KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Incentive Analysis in Permissionless Blockchains

Roozbeh Sarenche

Dissertation presented in partial fulfillment
of the requirements for the degree of
Doctor of Engineering Science (PhD):
Electrical Engineering

Supervisors:
Prof. dr. ir. Bart Preneel
Dr. Svetla Petkova-Nikova

September 2025

INCENTIVE ANALYSIS IN PERMISSIONLESS BLOCKCHAINS

Roozbeh SARENCHE

Supervisors:
Prof. dr. ir. Bart Preneel
Dr. Svetla Petkova-Nikova

Members of the
Examination Committee:
Prof. em. dr. ir. Jean-Pierre Celis, chair
Prof. dr. ir. Frederik Vercauteren
Prof. dr. ir. Wouter Joosen
Prof. dr. Tom Van Cutsem
Prof. dr. Joseph Bonneau
(New York University)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

September 2025

© 2025 Roozbeh Sarenche
Uitgegeven in eigen beheer, Roozbeh Sarenche, Leuven (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm, electronic or any other means without written permission from the publisher.

Acknowledgements

This thesis is the story of my last five years, years marked by both challenges and joys. Although my PhD journey formally concludes with these pages, I see this not as an end but as a new beginning in my life, enriched with valuable experiences.

First and foremost, I would like to express my deepest gratitude to my supervisors, Bart and Svetla. Throughout these years, they have always been supportive and never lost trust in me. They gave me the freedom and space to explore my ideas, while guiding me whenever I needed it. Their encouragement has been a source of confidence, and without their trust, this work would not have been possible.

I would also like to sincerely thank my chairman, Jean-Pierre Celis, as well as my committee members, Frederik Vercauteren, Wouter Joosen, Tom Van Cutsem, and Joseph Bonneau. Their valuable comments and feedback have significantly improved this thesis and broadened my perspective. I am grateful for the time and attention they devoted to evaluating my work.

A very special thank you goes to my wife, Samaneh. Pursuing my PhD meant leaving Iran for Belgium, which kept us apart for the first three years. Despite the distance, she continuously supported me and loved me unconditionally. When we were finally reunited, her presence gave me the strength to complete this journey. Thank you for always believing in me and trusting me, even when the circumstances were not easy.

I owe everything to my parents, to whom I dedicate this work with all my heart. It has not been easy for them to live far from their beloved child, but they have sacrificed so much for my success. From my very first years at school until this point, I have never felt without support. They were always there, providing the best for me. From a small city in Iran, Zanjan, to the best universities in Iran and Europe—this path would not have been possible without them.

I also want to thank my brother, Reza, and my sisters, Shiva and Shabnam. I have always been your little brother, taken care of and loved in every situation.

While writing this thesis, my siblings in Iran went through very difficult times. I hope that one day, when I look back at this work, I will see that we have overcome those hardships together. Be strong, my beloved family. I miss you all deeply, including my lovely nephew and nieces——Ariyan, Barsam, Rayan, and Taraneh. I wish you could be here to attend my defense.

Finally, I would like to thank all my friends and colleagues at COSIC. It has been a wonderful place where I could always feel peace and comfort. I will always remember the joy of having lunch together with Alma fans, the laughter during ping-pong matches, and the fun of playing padel. These moments of friendship and lightness gave me balance throughout my PhD journey. And to close, I would like to extend my special thanks to the COSIC administrative team——Pela, Dana, Saartje, and Elsy——who always made our work as students much easier.

Roozbeh

Popularized Abstract

Blockchains, the technology behind cryptocurrencies like Bitcoin and Ethereum, are designed to function without the interference of central authorities such as banks. Instead, they rely on thousands of independent participants across the globe to keep the system running. To ensure that these participants cooperate and engage in the protocol, which is necessary for the progress of the blockchain, the system rewards them with cryptocurrency for their efforts. However, as in any system based on rewards, there is always the temptation to "cheat the rules" in order to earn more than one's fair share.

This thesis explores how the design of blockchain reward mechanisms can sometimes open the door for such unfair strategies. In these situations, a small group of participants may be able to bend the rules, gaining more rewards than the resources they actually contribute. This not only gives them an advantage but also discourages honest participants, who may eventually stop contributing altogether. If that happens, the security and stability of the entire blockchain could be at risk. Understanding these vulnerabilities is therefore essential for both researchers and practitioners who seek to build trustworthy decentralized systems.

By studying major blockchains such as Bitcoin, Cardano, and Ethereum, this work shows where current reward systems fall short and how they can be improved. The broader goal is to help make blockchains fairer and more robust so that they can continue to support new applications without being undermined by malicious participants. Although prominent blockchains have not yet been victims of long-lasting, detectable incentive attacks from insiders, we cannot take this as a guarantee of their future safety. This highlights the need to enhance blockchain incentive structures to mitigate future threats more effectively.

Gepopulariseerde samenvatting

Blockchains, de technologie achter cryptovaluta zoals Bitcoin en Ethereum, zijn ontworpen om te functioneren zonder inmenging van centrale autoriteiten zoals banken. In plaats daarvan vertrouwen zij op duizenden onafhankelijke deelnemers verspreid over de hele wereld om het systeem draaiende te houden. Om ervoor te zorgen dat deze deelnemers samenwerken en deelnemen aan het protocol, wat noodzakelijk is voor de vooruitgang van de blockchain, beloont het systeem hen met cryptovaluta voor hun inspanningen. Zoals in elk systeem dat op beloningen is gebaseerd, bestaat echter altijd de verleiding om de regels te "omzeilen" om zo meer te verdienen dan een eerlijk aandeel.

Deze thesis onderzoekt hoe het ontwerp van beloningsmechanismen in blockchains soms de deur kan openen voor zulke oneerlijke strategieën. In die situaties kan een kleine groep deelnemers de regels buigen en meer beloningen verkrijgen dan overeenkomt met de middelen die zij daadwerkelijk bijdragen. Dit geeft hen niet alleen een voordeel, maar ontmoedigt ook eerlijke deelnemers, die uiteindelijk misschien helemaal stoppen met bijdragen. Als dat gebeurt, kunnen de veiligheid en stabiliteit van de gehele blockchain in gevaar komen. Het begrijpen van deze kwetsbaarheden is daarom essentieel voor zowel onderzoekers als praktijkmensen die betrouwbare gedecentraliseerde systemen willen bouwen.

Door grote blockchains zoals Bitcoin, Cardano en Ethereum te bestuderen, laat dit werk zien waar huidige beloningssystemen tekortschieten en hoe ze verbeterd kunnen worden. Het bredere doel is blockchains eerlijker en robuuster te maken, zodat ze nieuwe toepassingen kunnen blijven ondersteunen zonder te worden ondermijnd door kwaadwillende deelnemers. Hoewel prominente blockchains tot nu toe nog geen slachtoffer zijn geworden van langdurige, detecteerbare incentive-aanvallen van insiders, kunnen we dit niet zien als een garantie voor hun toekomstige veiligheid. Dit benadrukt de noodzaak om beloningsstructuren

binnen blockchains te versterken om toekomstige bedreigingen effectiever het hoofd te bieden.

Abstract

In recent years, permissionless blockchains have emerged as the foundation for a wide range of decentralized applications, including cryptocurrency management, decentralized finance, auctions, voting systems, digital ownership, and more. The reliability of these applications depends on the security guarantees offered by the underlying blockchain protocols. Although prominent permissionless blockchains often base their security on formal analyses and claims, these assessments typically rely on simplifying assumptions, such as the existence of an honest majority, or overlook critical factors, like the influence of reward distribution on system security.

Permissionless blockchains incentivize their participants—referred to as miners or validators—by rewarding them with cryptocurrency to encourage active participation in the protocol. The incentive mechanism plays a crucial role in maintaining the security of the blockchain by motivating participants to follow the desired, or honest, behavior. Since validators and miners are typically rational actors seeking to maximize profit, any flaw in the incentive design may motivate them to deviate from honest behavior in pursuit of an unfair advantage. Such exploitation of the incentive mechanism can lead to a destabilizing situation in which disadvantaged participants may choose either to deviate from honest behavior or to eventually abandon the network, with both possibilities threatening the blockchain's continued progress and stability.

In this thesis, we analyze the incentive mechanisms behind several prominent permissionless blockchains, including Bitcoin, Cardano, and Ethereum. Our focus is primarily on potential fairness attacks that allow adversarial miners or validators to increase their profits by capturing an unfair share of blocks. As a first step, we analyze the incentive structure in Bitcoin, the pioneering Proof-of-Work (PoW) blockchain. Specifically, we examine the temporal aspects of profitability associated with one of the most well-known fairness attacks, namely selfish mining, which aims to exclude honest blocks from the longest chain. Building on this, we assess the role of Bitcoin's difficulty adjustment mechanism in mitigating such attacks. We then explore the implications of mining pool

rationality for Bitcoin's security and propose novel attack vectors that could further increase an adversary's profit in a rational setting. Additionally, we investigate how upcoming Bitcoin reward halvings and the transition toward a transaction-fee era can impact Bitcoin's security by lowering the security threshold and increasing the potential for immediate profitability.

As a next step, we analyze the incentive mechanisms of Proof-of-Stake (PoS) protocols. We examine longest-chain PoS protocols, such as Cardano, and show that due to the block proposer predictability inherent in PoS systems, the selfish mining attack can be even more destructive in this context. We also analyze Ethereum as a voting-based PoS protocol. These protocols are susceptible to block reorganization attacks, in which an adversarial validator attempts to exclude blocks with weak voting support from the canonical chain. We identify a vulnerability in Ethereum's reward mechanism that results from the centralized distribution of vote rewards. If exploited, this vulnerability can lead to both short- and long-range reorganization attacks. To address this issue, we propose a fairer and more decentralized reward mechanism for Ethereum.

We hope the insights learned from this thesis will help identify potential incentive-based vulnerabilities in permissionless blockchains and contribute to the design of fairer blockchain protocols.

Beknopte samenvatting

In de afgelopen jaren zijn permissionless blockchains uitgegroeid tot de basis voor een breed scala aan gedecentraliseerde toepassingen, waaronder cryptovalutabeheer, gedecentraliseerde financiën, veilingen, stemsystemen, digitaal eigendom en meer. De betrouwbaarheid van deze toepassingen hangt af van de veiligheidsgaranties die worden geboden door de onderliggende blockchainprotocollen. Hoewel prominente permissionless blockchains hun veiligheid vaak baseren op formele analyses en beweringen, steunen deze beoordelingen doorgaans op vereenvoudigde aannames, zoals het bestaan van een eerlijke meerderheid, of negeren zij cruciale factoren, zoals de invloed van beloningsverdeling op de systeemveiligheid.

Permissionless blockchains stimuleren hun deelnemers — aangeduid als miners of validators — door hen te belonen met cryptovaluta, om actieve deelname aan het protocol aan te moedigen. Dit incentive-mechanisme speelt een cruciale rol bij het waarborgen van de veiligheid van de blockchain, doordat het deelnemers motiveert om het gewenste, eerlijke gedrag te volgen. Aangezien validators en miners doorgaans rationele actoren zijn die hun winst willen maximaliseren, kan elk gebrek in het incentive-ontwerp hen ertoe aanzetten af te wijken van eerlijk gedrag om een oneerlijk voordeel te behalen. Dergelijke uitbuiting van het incentive-mechanisme kan leiden tot een destabiliserende situatie waarin benadeelde deelnemers ervoor kiezen om af te wijken van eerlijk gedrag of uiteindelijk het netwerk te verlaten, waarbij beide mogelijkheden de voortgang en stabiliteit van de blockchain bedreigen.

In dit proefschrift analyseren we de incentive-mechanismen van verschillende prominente permissionless blockchains, waaronder Bitcoin, Cardano en Ethereum. Onze focus ligt voornamelijk op potentiële eerlijkheidsaanvallen waarmee kwaadwillende miners of validators hun winst kunnen verhogen door een onevenredig aandeel van de blokken te bemachtigen. Als eerste stap analyseren we de incentivestructuur van Bitcoin, de pionierende Proof-of-Work (PoW) blockchain. We onderzoeken in het bijzonder de temporele aspecten van winstgevendheid in verband met een van de bekendste eerlijkheidsaanvallen,

genaamd selfish mining, die tot doel heeft eerlijke blokken uit de langste keten uit te sluiten. Vervolgens beoordelen we de rol van Bitcoin's moeilijkheidsaanpassingsmechanisme bij het tegengaan van dergelijke aanvallen. Daarna verkennen we de implicaties van rationeel gedrag door mining pools voor de veiligheid van Bitcoin en stellen we nieuwe aanvalsvectoren voor die de winst van een tegenstander in een rationele setting verder kunnen vergroten. Daarnaast onderzoeken we hoe aankomende Bitcoin-halveringen en de overgang naar een tijdperk van transactiekosten de veiligheid van Bitcoin kunnen beïnvloeden door de veiligheidsdrempel te verlagen en het potentieel voor directe winstgevendheid te vergroten.

Vervolgens analyseren we de incentive-mechanismen van Proof-of-Stake (PoS) protocollen. We onderzoeken longest-chain PoS-protocollen, zoals Cardano, en tonen aan dat door de voorspelbaarheid van blokvoorstellers in PoS-systemen de selfish mining-aanval in deze context zelfs destructiever kan zijn. We analyseren ook Ethereum als een stemgebaseerd PoS-protocol. Deze protocollen zijn kwetsbaar voor reorganisatie-aanvallen, waarbij een kwaadwillende validator probeert blokken met zwakke stemondersteuning uit de canonieke keten uit te sluiten. We identificeren een kwetsbaarheid in het beloningsmechanisme van Ethereum die voortkomt uit de gecentraliseerde verdeling van stemvergoedingen. Indien uitgebuit, kan deze kwetsbaarheid leiden tot zowel korte- als langeafstand reorganisatie-aanvallen. Om dit probleem aan te pakken, stellen we een eerlijker en meer gedecentraliseerd beloningsmechanisme voor Ethereum voor.

We hopen dat de inzichten uit dit proefschrift zullen bijdragen aan het identificeren van potentiële incentive-gebaseerde kwetsbaarheden in permissionless blockchains en aan het ontwerpen van eerlijkere blockchainprotocollen.

List of Abbreviations

A3C	Asynchronous Advantage Actor-Critic
BDoS	Block Denial of Service
BFT	Byzantine Fault Tolerance
DAG	Directed Acyclic Graph
DAM	Difficulty Adjustment Mechanism
DeFi	Decentralized Finance
DLT	Distributed Ledger Technology
DoS	Denial of Service
DQL	Deep Q-Learning
FFG	Friendly Finality Gadget
ISM	Intermittent Selfish Mining
LC-PoS	Longest Chain Proof-of-Stake
LMD-GHOST	Latest Message Driven Greedy Heaviest Observed SubTree
MCTS	Monte Carlo Tree Search
MEV	Maximal (Miner) Extractable Value
OEA	Orphan Exclusion Attack
PoS	Proof of Stake
PoW	Proof of Work
SISM	Smart Intermittent Selfish Mining
SNE	Strong Nash Equilibrium
SP	Selfish Proposing
SPV	Simplified Payment Verification
UAV	Unmanned Aerial Vehicle

VRF Verifiable Random Function

List of Symbols

$\mathcal{A}$	Adversary
$\alpha_{\mathcal{A}}$	Adversarial mining/stake share
B	Block
$\mathtt{BR}$	Block ratio
λ	Bitcoin block mining rate
TGT	Bitcoin block/epoch target
d	Bitcoin difficulty adjustment damping factor
L	Bitcoin epoch block length
$\mathtt{Diff}$	Bitcoin block/epoch difficulty
Δ^{epoch}	Bitcoin epoch duration
$\mathtt{C}$	Chain
$\mathcal{H}$	Honest nodes
α	Mining/stake share
π	Mining/block proposing strategy
$\mathsf{DAM}^{\mathtt{modified}}$	Modified difficulty adjustment mechanism
Δ	Network delay upper bound
$l_{\mathcal{A}}$	Number of adversarial blocks
$l_{\mathcal{H}}$	Number of honest blocks
k	Number of recent blocks trimmed from a chain
N^{range}	Number of satoshi-per-virtual-byte ranges in the mempool
F	The normalized amount of extra fee of a whale block
t	Time/time slot
$T_{\mathtt{conf}}$	Transaction confirmation time

Contents

Popularized Abstract iii

Gepopulariseerde samenvatting v

Abstract vii

Beknopte samenvatting ix

List of Abbreviations xi

List of Symbols xiii

Contents xv

List of Figures xxv

List of Tables xxvii

I Prelude 1

1 Introduction 3

1.1 Motivation 3

1.2 Contributions 7

1.2.1 Incentive Analysis in Bitcoin 7

1.2.2 Incentive Analysis in PoS-Based Protocols 9

1.3 Outline of This Thesis 10

1.4 Other Publications 11

2 Preliminaries on Blockchain Security 13

2.1 Blockchain . . . 13
2.1.1 Degree of Permissionlessness in Blockchains . . . 13
2.2 Prominent Permissionless Blockchains . . . 14
2.2.1 Bitcoin . . . 14
2.2.2 Cardano . . . 16
2.2.3 Ethereum . . . 18
2.3 Security Analysis of Permissionless Blockchains . . . 20
2.3.1 Secure Blockchain Protocols . . . 20
2.3.2 Network Models . . . 21
2.3.3 The CAP Theorem . . . 21
2.3.4 Security Analysis of Longest Chain-Based Protocols: Bitcoin and Cardano . . . 22
2.3.5 Security Analysis of Ebb-and-Flow Protocols: Ethereum 22
2.4 Incentivization Mechanism in Permissionless Blockchains . . . 23
2.4.1 Analysis of the Incentivization Mechanism in Bitcoin . 25
2.4.2 Analysis of the Incentivization Mechanism in Cardano 26
2.4.3 Analysis of the Incentivization Mechanism in Ethereum 27
2.5 Selfish Mining: A Fairness Attack on Bitcoin . . . 28
2.5.1 Simple Selfish Mining Strategy in Bitcoin . . . 28
2.5.2 Optimal Selfish Mining Strategy in Bitcoin . . . 30
2.5.3 Mining Share Threshold for Profitable Selfish Mining . 30
2.5.4 Background on the Time-Averaged Profitability of Selfish Mining . . . 31
2.5.5 Intermittent Selfish Mining and Profit Lag . . . 33

3 Incentive Analysis in Bitcoin 35

3.1 Time-Averaged Analysis of Selfish Mining . . . 35
3.1.1 The Smart Intermittent Selfish Mining (Version 1) . . 36
3.1.2 Modified Difficulty Adjustment Mechanism . . . 37
3.1.3 Potential Attacks on Modified DAMs . . . 38
3.1.4 Effectiveness of Modified DAMs Against Selfish Mining 40
3.2 Security Analysis of Bitcoin in a Semi-Rational Setting . . . 41
3.2.1 Selfish Mining in a Semi-Rational Setting . . . 42
3.2.2 Bribery Attack . . . 43
3.2.3 Distraction Attack . . . 44
3.3 Bitcoin under Volatile Block Rewards . . . 46
3.3.1 Revisiting the Bitcoin Security Threshold . . . 46
3.3.2 Simplified Volatile Reward Model . . . 48
3.3.3 Enhanced Volatile Reward Model . . . 49

4 Incentive Analysis in PoS Blockchains **53**

4.1 Selfish Proposing in Longest-Chain Proof-of-Stake Protocols . 53
4.1.1 Selfish Proposing in LC-PoS Protocols with Perfect Randomness 54
4.1.2 Selfish Proposing in Full-Predictable LC-PoS 55
4.1.3 Deep Q-Learning for Analyzing Selfish Proposing . . . 56

4.2 Commitment Attacks on Ethereum's Reward Mechanism . . . 58
4.2.1 Commitment Attacks: Overview and Causes 58
4.2.2 New Vote Reward Mechanism 61

5 Conclusion and Future Work **63**

5.1 Conclusion 63

5.2 Future Work 64

Bibliography **75**

II Publications **77**

6 Selfish Mining Time-Averaged Analysis in Bitcoin **79**

6.1 Introduction 80

6.2 Related Work 83

6.3 Preliminaries 84
6.3.1 System Model and Definitions 84

6.4 Background on Selfish Mining Profitability 86

6.5 Intermittent Selfish Mining 89
6.5.1 Normal Intermittent Selfish Mining 89
6.5.2 Smart Intermittent Selfish Mining (Version 1) 91

6.6 Selfish Mining under the Modified DAM 94
6.6.1 The Uncle Blocks 95
6.6.2 Modified DAM with a Fixed Total Block Count per Epoch 96
6.6.3 Modified DAM with a Fixed Main-Chain Block Count per Epoch 96
6.6.4 Normal Selfish Mining under the Modified DAM . . . 97

6.7 Smart Intermittent Selfish Mining (Version 2) 99
6.7.1 Analysis under $\mathsf{DAM}_1^{\texttt{modified}}$ 100
6.7.2 Analysis under $\mathsf{DAM}_2^{\texttt{modified}}$ 101
6.7.3 Profitability of SISM2 under the Modified DAM . . . 101

6.8 Orphan Exclusion Attack . . . 103
6.8.1 Attack Description . . . 103
6.8.2 The Length of the Orphan Exclusion Attack . . . 105
6.9 Selfish Mining Accompanied by the Orphan Exclusion Attack 107
6.9.1 Analysis under $\mathsf{DAM}_1^{\mathtt{modified}}$ . . . 108
6.9.2 Analysis under $\mathsf{DAM}_2^{\mathtt{modified}}$ . . . 109
6.9.3 Profitability of Normal Selfish Mining Accompanied by the Orphan Exclusion Attack under the Modified DAM 110
6.10 Discussion . . . 110
6.10.1 Selfish Mining Countermeasures . . . 111
6.10.2 Reporting Orphan Blocks: Can It Solve All the Problems? 114
6.10.3 Future Work . . . 115
6.11 Conclusion . . . 116
Bibliography . . . 117

7 Mining Power Destruction Attacks 123

7.1 Introduction . . . 124
7.2 Preliminaries and System Model . . . 128
7.3 Selfish Mining Attack . . . 130
7.3.1 Theoretical Analysis . . . 131
7.4 Bribery Attack . . . 134
7.4.1 Bribery Attack Under the Assumption of Known Miners 136
7.5 Discussion . . . 139
7.6 Conclusion . . . 141
Bibliography . . . 142
7.A1 Parameters Involved in Semi-Rational Fork Choice . . . 147
7.A2 Our System Model Limitations . . . 147
7.A3 Selfish Mining Attack: Supplementary . . . 148
7.A3.1 Whale Transactions Used as Bribes . . . 148
7.A3.2 Proof of Lemma 7.1 . . . 148
7.A3.3 Selfish Mining Strategy . . . 149
7.A3.4 Proof of Lemma 7.2 . . . 151
7.A3.5 Proof of Thm. 7.1 . . . 156
7.A3.6 Proof of Cor. 7.1 . . . 157
7.A3.7 Mining Pools . . . 157
7.A3.8 MDP-Based Analysis . . . 158
7.A3.9 Smart Contract-Based Selfish Mining . . . 159

7.A3.10 Duration and Cost of the Initial Loss Period in Selfish Mining . . . 161
7.A4 Bribery Attack: Supplementary . . . 163
7.A4.1 Proof of Thm. 7.2 . . . 163
7.A4.2 Bribery Attack under the Assumption of Unknown Miners 164
7.A4.3 Bribery Attack Using Whale Transactions . . . 165
7.A4.4 Proof of Thm. 7.3 . . . 168
7.A4.5 Markov Chain Analysis of Bribery Attack . . . 170
7.A4.6 Markov Chain Analysis of Undercutting Attack . . . 172
7.A4.7 Naive Solutions to Mitigate the Bribery Attack . . . 175
7.A4.8 Smart Contract-Based vs. Whale Transaction-Based Bribery Attacks . . . 175
7.A4.9 The Impact of DAM on the Profitability of Selfish Mining and Bribery Attacks . . . 177
7.A5 Mining Power Distraction Attack . . . 178
7.A5.1 Expected Return Calculations . . . 181
7.A5.2 Quantitative Analysis of the Distraction Attack . . . 184
7.A5.3 BDoS Attack Description . . . 185
7.A5.4 Revealing Transactions . . . 185
7.A5.5 Distraction Attack Using the Non-Adversarial Blocks . 186

8 Bitcoin under Volatile Block Rewards 187

8.1 Introduction . . . 188
8.2 Preliminaries . . . 192
8.2.1 System Model and Definitions . . . 192
8.2.2 Overview of Selfish Mining Profitability . . . 195
8.2.3 Asynchronous Advantage Actor-Critic (A3C) . . . 196
8.3 Revisiting the WeRLman Model . . . 198
8.3.1 Rationale for Higher Security Threshold in Non-Predictable WeRLman . . . 200
8.4 Simplified Volatile Reward Model . . . 202
8.4.1 Block Generation Time Variability and Reward Volatility 203
8.4.2 Simplified Mempool Environment . . . 203
8.4.3 MDP-Based Tool . . . 204
8.4.4 Difficulty-Adjusted Selfish Mining in the Simplified Environment . . . 204
8.5 Pre-DAM Selfish Mining Profitability under Volatile Reward Model . . . 206
8.5.1 Selfish Mining Profit Lag under the Fixed Block Reward Model . . . 207

8.5.2 Profitability of Pre-DAM Selfish Mining under the Volatile Block Reward Model . . . 207
8.5.3 Pre-DAM Selfish Mining in the Simplified Environment 209
8.6 Enhanced Volatile Reward Model . . . 211
8.6.1 Enhanced Mempool Implementation . . . 213
8.6.2 A3C Actions and State Representation . . . 214
8.6.3 Undercutting as a Continuous Action . . . 217
8.6.4 Results and Implications: Mining Attacks under an Enhanced Volatile Model . . . 218
8.7 Conclusion . . . 221
Bibliography . . . 223
8.A1 Mining Attacks . . . 226
8.A1.1 Selfish Mining . . . 226
8.A1.2 Undercutting . . . 227
8.A2 Non-Predictable WeRLman . . . 228
8.A3 Adversarial Strategies in the Original and Non-predictable WeRLman Environments . . . 228
8.A3.1 Strategy π_1^{WeRLman} . . . 230
8.A3.2 Strategy $\pi_1^{\text{Non-predictable}}$ . . . 231
8.A3.3 Strategy $\pi_2^{\text{Non-predictable}}$ . . . 233
8.A4 Simplified Mempool Environment: Supplementary . . . 234
8.A5 Limitations of the Simplified Environment . . . 236
8.A6 A3C Implementation . . . 237
8.A6.1 Objective Function . . . 237
8.A6.2 Implementation Details . . . 240
8.A6.3 Implementation of the Undercutting Attack . . . 241
8.A7 Selfish Mining After Difficulty Adjustment . . . 241
8.A8 Selfish Mining Before Difficulty Adjustment: Supplementary . 244

9 Selfish Proposing in Longest-Chain Proof-of-Stake Protocols 247

9.1 Introduction . . . 248
9.2 Nothing-at-Stake Selfish Proposing in LC-PoS Protocols . . . 251
9.2.1 Intuition Behind Nothing-at-Stake Selfish Proposing . 252
9.2.2 Set of Actions for Nothing-at-Stake Selfish Proposing 253
9.2.3 Nothing-at-Stake Selfish Proposing with Perfect Randomness . . . 254

9.3 Selfish Proposing in Predictable LC-PoS Protocols 255
9.3.1 Selfish Proposing in Full-Predictable LC-PoS Protocols 256
9.4 Deep Q-Learning to Analyze Selfish Proposing 260
9.4.1 DQL Implementation in Blockchain Environments . . 260
9.4.2 DQL Implementation of Nothing-at-Stake Selfish Proposing 262
9.4.3 DQL Implementation of Full-Predictable Selfish Proposing 262
9.4.4 DQL Implementation of Semi-Predictable Selfish Proposing 263
9.5 Conclusion 264
Bibliography 265
9.A1 Preliminaries 268
9.A1.1 Selfish Mining in PoW 268
9.A1.2 Deep Q-Learning 268
9.A2 Proof of Stake 270
9.A2.1 Overview of an Existing PoS Model 270
9.A2.2 Formal Model to Define PoS Protocols 270
9.A2.3 Supplementary Definitions 275
9.A3 On Profitability of the Selfish Proposing Attack 276
9.A3.1 Block Ratio 277
9.A3.2 Relative Revenue 278
9.A3.3 Time-Averaged Profit 280
9.A4 Selfish Proposing Attack in LC-PoS Protocols with Perfect Randomness 283
9.A4.1 Strategy $\pi^{\text{S-PR}}$ 283
9.A4.2 Block Ratio of Strategy $\pi^{\text{S-PR}}$ 285
9.A5 Strategy $\pi^{\texttt{SP1}}$ in Full-Predictable Protocols 294
9.A6 Strategy $\pi^{\texttt{SP2}}$ in Full-Predictable Protocols 300
9.A7 Strategy $\pi^{\texttt{O-SP}}$ 305
9.A8 Justifying the Optimality of Strategy $\pi^{\texttt{O-SP}}$ 307
9.A9 Deep Q-Learning 308
9.A9.1 Implementation Details 308
9.A9.2 DQL Implementation Considering Node Locations . . 309

10 Commitment Attacks on Ethereum's Reward Mechanism 311

10.1 Introduction 312

10.1.1 Ethereum Consensus Security . . . 312
10.1.2 Incentives in Ethereum . . . 313
10.1.3 Commitment Attacks on LMD GHOST . . . 314
10.1.4 A New Reward Mechanism: DAG Votes . . . 317
10.2 Related Work . . . 319
10.2.1 Attacks on Ethereum Consensus . . . 319
10.2.2 Analogous Attacks on PoW Blockchains . . . 320
10.2.3 Timing Games . . . 320
10.2.4 Game Theory and Consensus Protocols . . . 321
10.3 Preliminaries . . . 321
10.3.1 Model . . . 321
10.3.2 The LMD GHOST Protocol . . . 322
10.3.3 The Ethereum Reward Mechanism . . . 324
10.4 The Simple Attack in the Presence of Solo Validators . . . 324
10.4.1 The Simple Game . . . 325
10.4.2 The Simple Game Analysis . . . 326
10.4.3 Credibility . . . 327
10.5 The Strong Simple Attack in the Presence of Solo Validators 328
10.5.1 The Strong Simple Game Analysis . . . 328
10.6 The Extended Attack in the Presence of Solo Validators . . . 330
10.6.1 The Extended Game . . . 330
10.6.2 The Adversarial Strategy . . . 330
10.6.3 Analysis . . . 332
10.7 Attacks in the Presence of Staking Pools . . . 333
10.7.1 The Simple Game . . . 333
10.7.2 The Extended Game . . . 334
10.7.3 Selfish Mining-Inspired Attack . . . 335
10.7.4 Analysis of Selfish Mining-Inspired Attack . . . 336
10.8 DAG Votes Mechanism . . . 337
10.8.1 Analysis of the DAG Votes Mechanism . . . 338
10.8.2 DAG Votes Made Practical . . . 340
10.8.3 Implementation of the DAG Votes . . . 342
10.9 Conclusion . . . 344
10.10 Responsible Disclosure . . . 344
10.11 Acknowledgment . . . 345
Bibliography . . . 346
10.A1 Extended Preliminaries . . . 353

10.A1.1 The Ethereum Reward Mechanism 353
10.A1.2 Slashing 354
10.A1.3 Reorgs 355
10.A2 Attack Quantification 356
10.A3 Proof of Theorem 10.3 357
10.A4 The Simple Attack in the Presence of Staking Pools 361
10.A5 The Extended Attack in the Presence of Staking Pools 362
10.A6 Performance Comparison: The Optimistic Scenario 363
10.A7 Performance Comparison: The Worst Case Scenario 365
10.A8 Impact of Future Ethereum Changes 366
10.A8.1 [block, slot]-voting 366
10.A8.2 Secret Leader Election (SLE) 367
10.A8.3 In-Protocol Proposer–Builder Separation (ePBS) . . . 367
10.A8.4 Single-Slot Finality 367
10.A9 Simple Attack without Proposer Boost 368

Statement on the Use of Generative AI **369**

Curriculum vitae **371**

List of Figures

4.1 A block path on a 2-D grid. 56

6.1 Time-averaged profit under the Bitcoin's DAM. 93
6.2 Profit lag under the Bitcoin's DAM ($\alpha = 0.27$). 94
6.3 Profit lag under the Bitcoin's DAM ($\alpha = \frac{1}{3}$). 94
6.4 Profitability advantage of SISM2 under $\mathsf{DAM}_1^{\texttt{modified}}$ 102
6.5 Profitability advantage of SISM2 under $\mathsf{DAM}_2^{\texttt{modified}}$ 103
6.6 Orphan exclusion attack 104
6.7 Comparison of orphan exclusion attacks: real-world vs. powerful attacker 106
6.8 The profitability advantage of $\pi^{\texttt{SM-OEA}}$ under the modified DAM 111

7.1 Mining share distribution between forks in a block race. 127
7.2 Periods of profitable selfish mining. 134
7.3 The adversarial reward share obtained from different attacks. . 139
7.4 The revenue advantage of the bribery attack over time. 140
7.5 The revenue advantage of selfish mining over time. 162
7.7 Quantitative analysis of the distraction attack. 184

8.1 Bitcoin security threshold before DAM. 191
8.2 Bitcoin security threshold in upcoming years. 200
8.3 Security threshold of adversarial strategies 201
8.4 Security threshold comparison under our simplified model and non-predictable WeRLman model. 207
8.5 The minimum number of days required for a selfish mining attack to become profitable under the fixed reward model. 208
8.6 Selfish mining under the volatile reward model. 209
8.7 Time-averaged profit before difficulty adjustment. 211
8.8 Total transaction weight for different ranges of sat/vByte as a function of block generation time (December 17, 2023). 214

8.9 The percentage increase in time-averaged profit (December 18, 2023; protocol reward: 3.125 BTC). 219
8.10 The percentage increase in time-averaged profit (December 18, 2023; protocol reward: 0 BTC). 220
8.11 The percentage increase in time-averaged profit (protocol reward: 0 BTC). 221
8.12 The profit percentage increase of undercutting. 222
8.13 Linear time-fee equation. 235
8.14 Time-fee equation. 237
8.15 The profit percentage increase of selfish mining after difficulty adjustment. 242
8.16 The concave time-fee function. 243
8.17 The percentage increase in time-averaged profit. 244

9.1 Nothing-at-stake selfish proposing 253
9.2 Block ratio of selfish proposing strategies in a full-predictable protocol 258
9.3 Block ratio comparison in a full-predictable protocol 259
9.4 Block ratio comparison 264
9.5 State representation of strategy $\pi^{\text{S-PR}}$ 285
9.6 Block path in Scenario 1 286
9.7 Representation of probability P_k 288
9.8 Block path in Scenario 2a 292
9.9 Block path in Scenario 2b 293
9.10 Block ratio comparison (communication capability equals 0.5.) 295
9.11 Block ratio comparison (communication capability equals 0.) . 295
9.12 Honest cycle of the strategy SP1 298
9.13 Adversarial cycle of the strategy SP1 299
9.14 Honest cycle of the strategy SP2 302
9.15 Adversarial cycle of the strategy SP2 304
9.16 Node locations 310

10.1 The simple game in the presence of non-colluding (solo) validators. 315
10.2 The extended attack. 316
10.3 The DAG votes reward mechanism. 318
10.4 Compliant blocks and votes in the extended game. 331
10.5 The payoff of a staking pool in the simple game. 334
10.6 The selfish mining-inspired attack. 335
10.7 MEV reward comparison under a successful and failed simple attack. 357
10.8 Subgame perfect Nash equilibrium 358

List of Tables

6.1 The length of the orphan exclusion attack 107

7.1 Mining Power Distribution of Bitcoin Mining Pools 157
7.2 Selfish mining in the presence of petty-compliant pools ($\alpha_{\mathcal{A}} = 0.4$). 160
7.3 Selfish mining in the presence of petty-compliant pools ($\alpha_{\mathcal{A}} = 0.3$). 160
7.4 Selfish mining results under real-world mining power distribution. 161
7.5 State transitions under the bribery attack. 172
7.6 State transitions under the undercutting attack. 174

8.1 Strategy π_1^{WeRLman}. 231
8.2 Strategy $\pi_1^{\text{Non-predictable}}$. 232
8.3 Strategy $\pi_2^{\text{Non-predictable}}$. 234

9.1 Block ratio comparison in a full-predictable protocol 260
9.2 States and their corresponding sequences 283
9.3 Strategy $\pi^{\text{S-PR}}$ 284

10.1 Pay-off matrix of a solo slot t attestor in the simple game. . . . 326
10.2 Expected payoff matrix of a solo slot t attestor in the strong simple game. 329
10.3 Pay-off matrix of a staking pool in the selfish mining-inspired game. 337
10.4 Comparison of Basic Ethereum Implementation and DAG Votes Mechanism 342
10.5 Payoff matrix of a solo slot $t = p$ validator in the extended game. 358
10.6 Payoff matrix of the slot $t = p$ leader in the extended game. . . 359
10.7 Payoff matrix of a solo slot $t = i$ validator in the extended game. 360
10.8 Payoff matrix of the slot $t = i$ leader in the extended game. . . 360
10.9 Payoff matrix of a staking pool in the simple game. 362

Part I

Prelude

Chapter 1

Introduction

1.1 Motivation

In 2008, Satoshi Nakamoto introduced the concept of blockchain as a unified ledger for recording transactions of a cryptocurrency named Bitcoin [74]. Since then, blockchain technology has become a foundational technology, giving rise to a diverse range of cryptocurrencies. Nowadays, blockchain technology has evolved into a well-established domain, earning widespread recognition not only among the general public due to the cryptocurrency hype but also within the research community for revolutionizing the concept of consensus in distributed systems. As the internet evolves toward the Web3 vision of decentralization, blockchains are expected to play a crucial role, serving as the backbone for a wide range of decentralized applications.

Prior to the introduction of Bitcoin, the research community had explored consensus protocols primarily through the topics of atomic broadcast and state machine replication [29, 45, 66, 67, 91]. The consensus protocols that emerged before Bitcoin, known as classical consensus protocols, were designed for a permissioned network, where all participating nodes are authorized, and the number of nodes is known in advance to all network participants. Although these permissioned consensus protocols were effective in designing distributed systems that required database replication, they faced limitations in achieving decentralization, where anyone can meaningfully contribute to the consensus process [69]. The desire for decentralized applications, such as digital currencies not governed by a central authority, drove the community to develop *permissionless* consensus protocols.

However, to make consensus protocols permissionless, several important challenges must be addressed. In a permissionless consensus protocol, anyone can participate, meaning that the identity and number of participants are unknown at the time of deployment and may vary over time. Determining the majority opinion without knowing the number of participants is a non-trivial task and poses a significant challenge for such consensus mechanisms. Additionally, in a permissionless setting, a single user may impersonate multiple participants to achieve a majority and alter the consensus outcome, a phenomenon known as Sybil attacks [82]. Permissionless consensus protocols should integrate a Sybil-resistance mechanism that distributes control not according to the number of nodes or identifiers a participant possesses, but based on the specific scarce resources they contribute. It turned out that it is impossible to achieve a permissionless consensus mechanism without resource restrictions [69].

The Sybil-resistance mechanism in Bitcoin's blockchain is called Proof-of-Work (PoW) [74], where the resource a participant controls is the computational power they own. The community has proposed several alternative Sybil-resistance mechanisms that are more energy-efficient and environmentally friendly than PoW. Examples include Proof-of-Stake (PoS) [63], where the resource is the cryptocurrency staked in the blockchain, and Proof-of-Space [16, 38], where the resource is the memory space owned by participants. Although a Sybil-resistance mechanism is necessary, it is not sufficient on its own to design permissionless consensus protocols. A consensus rule is also needed to determine the opinion of the participants who control the majority of resources. The community has proposed various consensus rules, ranging from the longest chain rule [74], where the accepted ledger is the one supported by the majority of available computational power, to classical voting-based schemes [26, 49], where the accepted ledger is the one receiving the greatest number of supporting votes. Another essential aspect of a permissionless blockchain is its incentivization mechanism. For a blockchain protocol to function effectively, a sufficient number of nodes must be motivated to actively participate in the protocol's operations. This incentivization is typically achieved by rewarding participants with a specified amount of native cryptocurrency for completing specific tasks.

According to the well-established definition of blockchain security in the literature, a blockchain protocol is considered secure if it satisfies two key properties: *consistency* and *liveness* [46, 47, 81, 93]. Roughly speaking, consistency ensures that all nodes in the blockchain network maintain a unified view of the blockchain ledger. Meanwhile, liveness guarantees that any well-formed transaction will eventually be included in the blockchain ledger. Security analyses of prominent blockchain technologies such as Bitcoin [46, 47], Ethereum [27, 77], and Cardano [17, 33, 62] indicate that, under specified network conditions and the honest majority assumption, these protocols can achieve the desired security properties.

The existence of a security proof for a blockchain protocol does not necessarily imply that the protocol is immune to all types of attacks, even under an honest majority. Several attacks [43, 87, 92] have been introduced in the literature where adversaries controlling a minority of the resource share exploit the incentivization mechanism to unfairly increase their reward share. While these attacks may not directly compromise the protocol's consistency or liveness, their practical impact can be severe, potentially leading to the collapse of the cryptocurrency and ultimately halting the protocol. A compromise in the incentivization mechanism can demotivate participants from engaging in the protocol, as they may find the rewards insufficient to compensate for the resources spent. Additionally, in large-scale permissionless blockchains, where most participants are rational and motivated by incentives, if they find that deviating from honest behavior is more profitable, the assumption of a majority adhering to the protocol is compromised, which in turn invalidates the security proof.

The importance of a well-designed incentivization mechanism in blockchain protocol security emphasizes the need for a protocol to ensure *fairness*, alongside the well-established properties of consistency and liveness. Fairness in a blockchain protocol means that a participant's reward share should be proportional to the resources they contribute [93]. The Bitcoin blockchain is inherently unfair due to its vulnerability to the *selfish mining* attack [43]. In this attack, an adversary controlling less than 50% of the computational power can undermine the Bitcoin blockchain by invalidating (orphaning) some honest blocks and replacing them with adversarial ones. Orphaning a block occurs when it is no longer part of the longest chain, causing it to be removed from the blockchain ledger. This attack results in an increase in the adversary's block ratio, and consequently, its reward share. Selfish mining is not exclusive to Bitcoin; it can also impact any longest-chain-based blockchain [22, 87], such as Cardano, which uses a PoS mechanism. Other permissionless blockchains that follow a voting-based mechanism, such as Ethereum and Solana, may also be vulnerable to fairness-targeting attacks, such as the *reorganization* attack [78, 92]. In this attack, the adversary attempts to invalidate honest blocks that receive a low number of supporting votes.

Permissionless blockchains, due to the rationality of their participants, are also vulnerable to *incentive manipulation* attacks [59]. These attacks aim to alter the incentives in a way that makes it profitable for non-adversarial but rational participants to deviate from the honest strategy. Examples of incentive manipulation attacks include bribery [21, 58, 70, 71], where participants are offered rewards to follow a prescribed adversarial strategy; block denial-of-service attacks [72], where the adversary incentivizes participants to stop engaging in the blockchain protocol; and distraction attacks [88, 95], where participants are motivated to shift to other platforms. These incentive manipulation attacks can

be applied either alone or in combination with attacks such as selfish mining, reorganization, and double spending, to increase the probability of success and maximize the adversary's profits.

Despite the theoretical possibility of attacks targeting the incentivization mechanism in blockchain protocols, prominent permissionless blockchains such as Bitcoin and Ethereum have not experienced long-lasting, detectable incentive attacks in recent years. While these blockchains have encountered block orphaning or reorganization, such events have generally been attributed to network issues, bugs, or incompatibilities in participants' software versions, rather than deliberate malicious attempts to compromise the blockchain. Attacks targeting large-scale blockchains typically require control of a significant amount of resources (not necessarily the majority, but at least a substantial share). This explains why solo participants or those with limited resources are unlikely to engage in such attacks. However, for wealthy entities controlling a large portion of resources, such as mining or staking pools, the most compelling reason to avoid these attacks is the potential negative impact on the price of the blockchain's native cryptocurrency [23]. While the adversary may earn more cryptocurrency through the attack, the consequences would leave the blockchain protocol in a scorched-earth state, leading to a collapse in the cryptocurrency's price.

None of the arguments, such as the lack of serious attacks on prominent blockchains thus far, the high resource requirements of such attacks, or their scorched-earth nature, guarantee that these attacks will not occur in the future. Some attacks have been introduced in the literature that require relatively low resources [88]. Additionally, future changes in the dynamics of incentivization mechanisms could make existing attacks possible with even lower resource shares. For example, studies have shown that Bitcoin's halving mechanism will reduce the minimum computational power required to conduct selfish mining in the future [19, 86].

We can also consider several scenarios that challenge the scorched-earth rationale for the absence of attacks, thereby strengthening the possibility of such attacks occurring in the future. One possibility is that blockchain protocols may face attacks that do not necessarily result in a scorched-earth outcome. Blockchain history has experienced deviations from honest strategies that did not lead to significant reputation damage, such as Ethereum validators outsourcing transaction payload creation [18, 55] or Bitcoin miners shifting computational power between Bitcoin and Bitcoin Cash [54, 65]. These examples suggest that if attacks are strategically executed or justified (e.g., claiming a specific block orphaning is due to compliance with sanction regulations), the negative impact on the cryptocurrency's price may be manageable or short-lived. Another possibility is that, at some point in the future, a significant portion of participants

may find participation in the protocol no longer worthwhile. In such a case, concerns about reputation or a scorched-earth outcome might become irrelevant, leading affected participants to opt for deviating from the honest strategy rather than leaving the protocol. A third scenario is an adversary intentionally attempting to destroy a blockchain in order to increase the market value of a competing platform.

Given the critical role of blockchain protocols in modern life, this thesis addresses the question of how the incentive mechanisms of permissionless blockchains, such as Bitcoin, Cardano, and Ethereum, can be exploited through attacks and what countermeasures can effectively mitigate these threats. To answer this, the objective is to provide an in-depth security analysis of these systems by evaluating existing attacks in practical scenarios, examining their implications under upcoming protocol changes, identifying new attack vectors, and proposing defenses to strengthen the security and resilience of blockchain protocols.

1.2 Contributions

The key contributions of this thesis can be categorized into two main research areas: i) incentive analysis of Bitcoin (a PoW-based protocol) and ii) incentive analysis of PoS-based protocols.

1.2.1 Incentive Analysis in Bitcoin

Our analysis of the Bitcoin protocol explores the impact of mining difficulty adjustment, mining pool rationality, and the increasing dominance of transaction fee rewards on Bitcoin's security.

C1. Time-averaged analysis of selfish mining in Bitcoin. In [90], we investigate the impact of Bitcoin's Difficulty Adjustment Mechanism (DAM)––which adjusts mining difficulty approximately every two weeks based on active computational power to maintain consistent block throughput––on the profitability of selfish mining strategies. Prior research shows that selfish mining involves an initial period of financial loss before the DAM adjusts (reduces) the mining difficulty. We introduce a novel *smart intermittent selfish mining attack* that shortens this initial loss period compared to existing variants. Furthermore, we challenge the belief that incorporating orphan blocks in DAM calculations completely eliminates selfish mining profitability. By analyzing two modified DAM versions that incorporate orphan blocks, we demonstrate that even these adaptations remain vulnerable to innovative attacks, such as the *orphan exclusion attack*, resulting in profitable selfish mining. Finally, we

discuss prominent selfish mining countermeasures and highlight their practical limitations.

C2. Analysis of mining power destruction attacks in Bitcoin under semi-rational settings. Most existing analyses of mining power destruction attacks in Bitcoin, including selfish mining, have simplified the environment by either ignoring rational participants (splitting miners into adversarial and honest groups) or by treating rational participants as solo nodes with negligible mining shares. In [88], we extend the analysis to a semi-rational setting, considering mining pools that can adapt their strategies in response to adversarial actions. By examining selfish mining in this context, we show that the distribution of mining power among pools and the gaps between smaller and larger pools can significantly affect Bitcoin's security. Additionally, we introduce a novel *bribery attack* that outperforms selfish mining for smaller pools, enabling any pool, regardless of its mining share, to target pools with smaller shares. Finally, we propose a novel *distraction attack*, where an adversary incentivizes mining pools to waste part of their mining power on a PoW puzzle defined outside the Bitcoin platform, leading to an increase in the adversary's profit without generating detectable evidence of an attack, such as orphan blocks.

C3. Analysis of the impact of transaction pool statistics on Bitcoin security in the transaction-fee era. The Bitcoin protocol features a halving mechanism that halves the protocol reward every four years. As a result, transaction fees are expected to become a major source of rewards in the near future, altering the dynamics of Bitcoin's incentivization mechanism. Existing research analyzing Bitcoin's security in the transaction-fee era often makes unrealistic assumptions, such as assuming unlimited block space in [28] or considering only two fee levels in [19]. In [86], we extend the security analysis to a more practical setting that incorporates the complexities of the Bitcoin mempool behavior. Our analysis emphasizes the impact of block generation time on block rewards, showing that blocks with longer generation times can be vulnerable to adversarial mining strategies. We revisit existing security thresholds for Bitcoin in the transaction-fee era, demonstrating that not only does the security threshold for a profitable deviation decrease, but the deviation can also become immediately profitable, eliminating the need to wait for difficulty adjustment. In [86], we also present a deep reinforcement learning tool that determines near-optimal Bitcoin mining strategies in the transaction-fee era, considering varying fee levels and replicating mempool patterns over any given period.

1.2.2 Incentive Analysis in PoS-Based Protocols

Our analysis of PoS-based blockchains involves both longest-chain-based blockchains, such as Cardano, and the leading voting-based blockchain, Ethereum.

C4. Analysis of selfish proposing attack in longest-chain PoS protocols. The selfish mining attack can target any longest-chain-based blockchain, regardless of its underlying Sybil resistance mechanism. In [87], we analyze the selfish proposing attack in the context of PoS protocols, such as Ouroboros [33, 62]—the consensus mechanism behind Cardano—and Snow White [30]. Our study covers various PoS environments: fully predictable PoS protocols, where the block proposer for a given slot is known to all participants in advance; semi-predictable PoS protocols, where the slot's proposer is determined by a Verifiable Random Function (VRF) and known only to the proposer beforehand; and idealized PoS protocols, where the slot's proposer is unknown to all participants until the slot begins, mimicking PoW behavior. We demonstrate that, due to the nothing-at-stake phenomenon and the predictability of block proposers in PoS protocols, the selfish proposing attack can be more destructive in any type of longest-chain PoS than in PoW. This conclusion is based on two key observations: first, with the same resource share in both longest-chain PoS and PoW protocols, the attack profit is higher in PoS protocols; and second, the security threshold for profitable deviation in practical PoS protocols diminishes to a negligible stake share.

C5. Analysis of commitment attacks on LMD-GHOST, the fork-choice rule of Ethereum. In [89], we introduce a series of commitment attacks on Ethereum's fork-choice rule, LMD-GHOST, where a malicious block proposer uses credible threats to incentivize rational voters from previous slots into adopting the adversarial strategy. These attacks range from reorging a single block to undermining the liveness of Ethereum's consensus protocol. The root cause of these attacks is a vulnerability in Ethereum's reward mechanism, which grants the proposer of the next slot full control over deciding whether a vote from a current slot qualifies for a reward (specifically, whether it was published on time). To address this vulnerability, we propose a novel reward mechanism that distributes decision-making power among a set of selected validators, ensuring greater fairness and decentralization. We prove the existence of a strong Nash equilibrium (SNE) such that LMD-GHOST equipped with our reward mechanism remains secure, even with rational voters and potentially adversarial block proposers. Furthermore, we demonstrate the feasibility of our proposed solution in a synchronous setting using a homomorphic signature scheme.

1.3 Outline of This Thesis

This thesis consists of two parts: *Prelude* (Part I) and *Publications* (Part II). The first part of this thesis includes the following chapters.

Chapter 2 provides an overview of the preliminaries and fundamental concepts in the analysis of permissionless blockchains. It briefly introduces the underlying Sybil resistance, consensus rules, and incentivization mechanisms employed by prominent permissionless blockchains, including Bitcoin, Cardano, and Ethereum. Chapter 2 also includes a brief overview of the selfish mining attack. Chapter 3 presents our contributions C1, C2, and C3 concerning the security analysis of Bitcoin's incentive mechanism. Chapter 4 discusses our contributions C4 and C5 related to the security analysis of incentive mechanisms in PoS blockchains.

Part II of this thesis contains the original text of our papers, listed as follows:

International Journals

- Chapter 6 (p. 79):

  Roozbeh Sarenche, Ren Zhang, Svetla Nikova, and Bart Preneel, "Selfish Mining Time-Averaged Analysis in Bitcoin: Is Orphan Reporting an Effective Countermeasure?," *IEEE Transactions on Information Forensics and Security*, volume 20, pages 449–464, 2024 (https://doi.org/10.1109/TIFS.2024.3518090).

International Conferences

- Chapter 7 (p. 123):

  Roozbeh Sarenche, Svetla Nikova, and Bart Preneel, "Mining Power Destruction Attacks in the Presence of Petty-Compliant Mining Pools," to appear in *International Conference on Financial Cryptography and Data Security (FC)*, 2025 (https://arxiv.org/abs/2502.07410).

- Chapter 8 (p. 187):

  Roozbeh Sarenche, Alireza Aghabagherloo, Svetla Nikova, and Bart Preneel, "Bitcoin under Volatile Block Rewards: How Mempool Statistics Can Influence Bitcoin Mining," to appear in *ACM Conference on Computer and Communications Security (CCS)*, 2025 (https://arxiv.org/abs/2411.11702).

- Chapter 9 (p. 247):

  Roozbeh Sarenche, Svetla Nikova, and Bart Preneel, "Deep Selfish Proposing in Longest-Chain Proof-of-Stake Protocols," in *International Conference on Financial Cryptography and Data Security (FC)*, volume 14744, pages 24–40, Springer, 2024 (https://eprint.iacr.org/2024/622).

- Chapter 10 (p. 311):

  Roozbeh Sarenche, Ertem Nusret Tas, Barnabé Monnot, Caspar Schwarz-Schilling, and Bart Preneel, "Commitment Attacks on Ethereum's Reward Mechanism," in *European Symposium on Security and Privacy (EuroS&P)*, pages 504–526, IEEE, 2025 (https://arxiv.org/abs/2407.19479).

1.4 Other Publications

Additionally, we have published several other articles that are not included in this thesis:

- Alireza Aghabagherloo, Roozbeh Sarenche, Maryam Zarezadeh, Bart Preneel, and Stefan Köpsell, "Priv-PFL: A Privacy-Preserving and Efficient Personalized Federated Learning Approach," in *IEEE Symposium on Security and Privacy Workshops (SPW)*, 2025 (https://eprint.iacr.org/2025/703).
  Summary. In this paper, we propose a privacy-preserving and efficient personalized federated learning framework that enhances robustness against poisoning attacks. We leverage fully homomorphic encryption to enable server-assisted detection of unpoisoned clients without requiring direct model sharing, thereby safeguarding client privacy. Furthermore, we introduce an efficient client selection algorithm that reduces overhead and ensures reliable detection of unpoisoned clients.

- Roozbeh Sarenche, Farhad Aghili, Dave Singelée, and Taka Yoshizawa, "DASLog: Decentralized Auditable Secure Logging for UAV Ecosystems," *IEEE Internet of Things Journal*, volume 10, pages 20264–20284, 2023 (https://doi.org/10.1109/JIOT.2023.3281263).
  Summary. In this paper, we introduce DASLog, a secure logging framework for Unmanned Aerial Vehicle (UAV) ecosystems, providing public auditability under single-operator control through a private blockchain for tamper-resistant storage of cryptographic proofs. Our design is optimized for efficient, high-speed logging, ensuring scalability for real-time UAV operations. A proof-of-concept implementation

using Hyperledger Besu [12] on cloud infrastructure validates DASLog's practicality for real-world deployment.

- Mahdi Montakhabi, Shenja Van der Graaf, Akash Madhusudan, Roozbeh Sarenche, and Mustafa Mustafa, "Fostering Energy Transition in Smart Cities: DLTs for Peer-to-Peer Electricity Trading," in *International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 466–472, IEEE, 2021 (https://doi.org/10.1109/DCOSS52077.2021.00078).
  Summary. In this paper, we explore the potential of Distributed Ledger Technologies (DLTs) to enable peer-to-peer electricity trading in the context of smart city energy transitions. We evaluate several DLTs by analyzing their performance and highlighting the trade-offs in scalability, privacy, security, and practicality. Our study provides insights into how different DLTs can support sustainable and efficient energy markets within future smart urban ecosystems.

Chapter 2

Preliminaries on Blockchain Security

2.1 Blockchain

A blockchain protocol enables a network of distributed nodes to achieve consensus over an ordered list of messages. The ordered list is typically called the blockchain *ledger*, and the messages are referred to as *transactions*. To improve transaction throughput (the number of transactions processed per unit of time), blockchain consensus protocols are applied at a time to a group of transactions, batched together into a unit called a *block*. These blocks are then sequentially linked together to form a *chain*.

Due to network delays, blockchain nodes may have different *views* of available chains. At each stage of the protocol, the blockchain nodes should select the *best* chain in their view and try to extend it. Each blockchain protocol defines specific *consensus rules* for determining the best chain. Intuitively speaking, the best chain is the one that the majority agrees is the most up-to-date ledger. We may use the terms *best* chain, *canonical* chain, and *main* chain interchangeably.

2.1.1 Degree of Permissionlessness in Blockchains

The core principle of *permissionless* blockchains is to establish an agreement on a shared ledger in a decentralized network, allowing anyone to participate without prior authorization from a central authority [34, 75]. As discussed by Lewis-Pye and Roughgarden [69], we can define a hierarchy of permissionlessness for

blockchain protocols. As the level of permissionlessness decreases, the protocol's knowledge of its participants increases, making it easier to design a secure protocol.

Fully permissionless setting: At each point in time, the protocol has no knowledge of the participating nodes.

Dynamically available permissionless setting: At each point in time, the protocol is aware of a dynamically available list of participants, where a subset of these participants is actively taking part in the protocol.

Quasi-permissionless setting: At each point in time, the protocol is aware of a dynamically available list of participants and assumes that all listed participants are actively taking part in the protocol.

Permissioned setting: The list of participants is fixed at the time of protocol deployment. At each point in time, the protocol assumes all participants are actively taking part in the protocol.

Any permissionless blockchain, in addition to consensus rules, requires a *Sybil resistance mechanism* to prevent a single actor from impersonating multiple participants. This is achieved by distributing power among participants in proportion to their share of a specific scarce resource.

2.2 Prominent Permissionless Blockchains

In this section, we briefly introduce the Sybil resistance mechanism and the consensus rule of three prominent permissionless blockchains: Bitcoin, Cardano, and Ethereum.

2.2.1 Bitcoin

Bitcoin's consensus mechanism relies on two key components: a Proof-of-Work (PoW)-based Sybil resistance mechanism for selecting block proposers and the longest-chain fork choice rule for determining the best chain [74]. According to the permissionlessness hierarchy introduced in Section 2.1.1, Bitcoin is classified as a fully permissionless blockchain.

2.2.1.1 The PoW-Based Sybil Resistance Mechanism of Bitcoin

In the Bitcoin PoW mechanism, the resource a participant controls is the computational power it owns. The PoW mechanism functions as a lottery that selects a node from the network to propose the next block. A node's likelihood of

winning this lottery is proportional to its computational resources. To precisely describe Bitcoin's PoW mechanism, we first outline the structure of a Bitcoin block.

Bitcoin Block Structure. A Bitcoin block consists of two key components: a block header and a list of transactions. The block header contains the metadata required to verify the block's validity, where its most important segments are as follows [4]: (1) the hash of the previous block: indicating the hash of the most recent block in the chain being extended by the current block; (2) the transactions' Merkle root: a condensed cryptographic proof of the transactions included in the block; (3) the timestamp: the approximate creation time of the block; (4) the difficulty target: representing the target difficulty that the current block hash has satisfied; and (5) the nonce: serving as a random number to introduce randomness to the block hash.

According to the Bitcoin PoW mechanism, the next block proposer is the first Bitcoin node to propose a block whose hash is smaller than the *difficulty target* specified by the protocol. More specifically, to propose the next block, each Bitcoin node first selects the best chain available in its view and then creates random candidate blocks on top of it. A candidate block includes a list of transactions that are valid with respect to the best chain, along with a header constructed using information from the best chain and the transaction list. If the candidate block's hash meets the difficulty target, the Bitcoin node broadcasts it as the next valid block to other nodes; otherwise, the node continues generating new candidate blocks. To generate new candidate blocks, a Bitcoin node can iterate through different nonce values in the block header, adjust the block timestamp, or alter the content or order of the included transactions. This process of finding a valid candidate block whose hash is less than the difficulty target is called *mining*. The term *miner* is also used to refer to a Bitcoin node that participates in this process.

2.2.1.2 The Fork Choice Rule of Bitcoin

To discuss Bitcoin's fork choice rule, we first review the concept of a chain's accumulated difficulty. The difficulty of a block B is defined as the ratio $\frac{\mathsf{TGT}^{\max}}{\mathsf{TGT}(B)}$, where $\mathsf{TGT}^{\max}$ denotes the maximum target (corresponding to the easiest allowable block), and $\mathsf{TGT}(B)$ is the target of block B. This ratio indicates how many times more difficult it is to mine block B compared to mining a block with the maximum target. The accumulated difficulty of a blockchain is then defined as the sum of the individual difficulties of all blocks in the chain [3].

According to the Bitcoin fork choice rule, when multiple forks are present in the view of a Bitcoin node, the node selects the one with the highest accumulated

difficulty. This rule is commonly referred to as the *longest chain* rule. This is because, if the diverging point of the forks is not too deep, the blocks in the differing forks typically have the same or nearly the same difficulty, implying that the longest fork should have the highest accumulated difficulty. If multiple forks have the same amount of accumulated difficulty, the Bitcoin node should select the one it received first in its view.

2.2.1.3 Bitcoin Difficulty Adjustment Mechanism

To ensure that the block generation rate, and consequently, the transaction throughput, remains at a constant level, the Bitcoin blockchain is equipped with a Difficulty Adjustment Mechanism (DAM) to determine the mining difficulty target. The purpose of DAM is to balance the Bitcoin mining difficulty with the active mining power available in the network, ensuring that a new Bitcoin block is mined in approximately 10 minutes.

In Bitcoin, a period during which 2016 consecutive blocks are added to the canonical chain is referred to as an *epoch*. At the end of each epoch, the DAM calculates the difficulty target for the upcoming epoch based on the estimated hash power of the previous epoch. At the end of epoch i, the difficulty target of epoch $i+1$, denoted by TGT_{i+1}, can be obtained as follows [90, 100]:

$$\mathsf{TGT}_{i+1} = \begin{cases} \mathsf{TGT}_i \cdot \frac{1}{d}, & \Delta_i^{\text{epoch}} < \frac{1}{d} \cdot \Delta_{\text{ideal}}^{\text{epoch}} \\ \mathsf{TGT}_i \cdot d, & \Delta_i^{\text{epoch}} > d \cdot \Delta_{\text{ideal}}^{\text{epoch}} \\ \mathsf{TGT}_i \cdot \frac{\Delta_i^{\text{epoch}}}{\Delta_{\text{ideal}}^{\text{epoch}}}, & \text{otherwise} \end{cases}, \tag{2.1}$$

where TGT_i is the difficulty target of epoch i, $\Delta_{\text{ideal}}^{\text{epoch}}$ is the ideal time duration of a Bitcoin epoch, which is equal to 2 weeks[1] , d is a dampening filter to prevent rapid changes in the mining difficulty and is set to $d = 4$ in the Bitcoin protocol, and Δ_i^{epoch} is the actual time duration of the last epoch, which can be calculated as the difference between the timestamps of the first and last blocks in epoch i.

2.2.2 Cardano

Cardano's consensus mechanism, called Ouroboros [17, 33, 62], employs a Proof-of-Stake (PoS)-based Sybil resistance mechanism and follows the same fork choice rule as Bitcoin, namely longest-chain. According to the permissionlessness

[1]The duration of the ideal Bitcoin epoch, consisting of 2016 blocks each with an average generation time of 10 minutes, is two weeks.

hierarchy introduced in Section 2.1.1, Cardano, similar to any longest-chain PoS protocol, operates in a dynamically available permissionless setting.

2.2.2.1 The PoS-Based Sybil Resistance Mechanism of Cardano

In a PoS-based blockchain, participants must deposit a certain amount of native cryptocurrency (stake) as their resource share in the blockchain ledger to take part in the protocol. A participant's chance of being selected as a block proposer is proportional to its deposited stake. More specifically, Cardano employs a Verifiable Random Function (VRF)-based lottery mechanism for block proposer selection. Each participant (also called a validator) who owns a deposit account generates a pair of VRF keys (a secret key and a verification key) associated with its account. Cardano divides time into one-second slots, and in each slot, every validator runs the VRF on the inputs: slot ID, its VRF secret key, and a random seed extracted from the blockchain ledger[2]. The VRF outputs a random number along with a proof, which can be verified using the validator's publicly known verification key.

If the random number is below a protocol-specified threshold, the validator becomes eligible to generate a block in that slot, and the block must include the VRF outputs as proof of proposer eligibility. This threshold is set per validator in proportion to the stake deposited in the linked account. This ensures that the probability of selection as a block proposer matches the participant's stake share. A slot in Cardano may remain empty or have multiple block proposers, with the threshold parameter adjusted to achieve an average of one block per 20 slots [6, 7].

2.2.2.2 The Fork Choice Rule of Cardano

The current version of Cardano's consensus protocol, Ouroboros Praos [33], follows the longest-chain fork choice rule, similar to Bitcoin. In the context of PoS blockchains, the longest chain refers to the one with the highest number of blocks. In the case of forks of equal length, the protocol selects the one whose final block has the lowest VRF value.

The longest-chain PoS protocols have been shown to be vulnerable to long-range attacks [35]. In such an attack, an adversary obtains the private keys of past validators who are no longer active in the network, allowing the adversary to fork the chain far back in history and privately build an alternative version.

[2]To ensure all participants use the same random seed, the random seed for a Cardano epoch is derived from the blockchain up to 2/3 of the previous epoch. As an epoch in Cardano consists of 432,000 slots, equivalent to five days, each epoch's random seed is known to participants well before the epoch starts.

With these keys, the adversary can simulate a majority stake in the private fork, which allows the fork to progress as if it had sufficient validator support. Over time, the private chain surpasses the public one in length or weight, enabling the adversary to overwrite the legitimate blockchain history.

Ouroboros Praos addresses this issue by enforcing a checkpointing functionality that prevents a Cardano validator from rolling back the best chain in its view by more than a security parameter of k blocks. However, this solution has a drawback in the dynamically available setting as new joining participants require trusted advice to determine the checkpointed chain.

A potential update to Ouroboros, called Ouroboros Genesis [17], aims to resolve this trust issue by modifying the longest-chain rule. If the fork divergence is at most k blocks, the best chain is the longer one. However, if the forks diverge for more than k blocks, the best chain is the denser one, defined as the chain with the fewest empty slots in a predefined time window that immediately starts after the forking point. This modification enhances resistance to long-range attacks, as immediately after such an attack begins, the active share in the adversarial private chain is lower than in the public chain, resulting in a higher occurrence of empty slots in the adversarial chain.

2.2.3 Ethereum

After the Merge in September 2022 [10], the Ethereum protocol transitioned to a PoS-based Sybil resistance mechanism. It employs two sub-protocols: the fork choice rule, known as the Latest Message Driven Greedy Heaviest Observed Sub-Tree (LMD-GHOST) [40], which outputs the best chain, and the Casper Friendly Finality Gadget (Casper FFG) [26], which checkpoints blocks on the best chain. Based on the permissionlessness hierarchy introduced in Section 2.1.1, the Casper FFG protocol operates in a quasi-permissionless setting. However, Ethereum is equipped with LMD-GHOST, enabling it to also operate in a dynamically available setting. More details on how Ethereum uses both Casper FFG and LMD-GHOST are provided in Section 2.3.5.

2.2.3.1 The PoS-Based Sybil Resistance Mechanism of Ethereum

Similar to Cardano, Ethereum employs a PoS-based lottery mechanism to determine block proposers. In Ethereum, time is divided into 12-second slots, with one validator randomly selected per slot to propose a block. An epoch in Ethereum consists of 32 slots. Ethereum features a decentralized random number generation system called RANDAO, which generates a random seed for each epoch used to select block proposers for the epoch's slots. Specifically, to determine the random seed for epoch i, the RANDAO system aggregates

random values submitted in each block of epoch $i-2$ to obtain the seed for epoch i. To assign block proposers, the Ethereum protocol randomly shuffles the validator set using the epoch seed and designates one validator per slot as its proposer. The number of tickets each validator holds in this lottery is proportional to its deposited stake [9].

2.2.3.2 The Fork Choice Rule of Ethereum

Ethereum's fork-choice rule, known as LMD-GHOST, can be seen as a variation of Bitcoin's longest-chain rule. In Bitcoin, when two forks compete, the valid chain is the one with the most accumulated PoW. In Ethereum, the *heaviest* chain is chosen in an analogous way, but weight is defined by validator votes rather than PoW. Informally, heaviest means the chain that has accumulated the most votes from validators, with only each validator's latest vote counting. The LMD-GHOST protocol is the mechanism that tallies these votes and walks down the chain starting from the latest justified block, moving step by step and always following the branch with the greatest weight until it reaches the tip.

At the start of each slot, the designated block proposer must run the LMD-GHOST protocol to select the best chain on which to propose a new block. In addition to the block proposer, each slot in Ethereum is assigned a committee of validators, known as attestors, who are randomly selected by the protocol to vote for the block at the tip of the best chain in their view. These votes help the next block proposer identify the best chain. Attestors also use the LMD-GHOST protocol to determine the best chain for voting. The LMD-GHOST protocol operates as follows [42, 92]: The protocol begins by selecting the latest justified block, as determined by Casper FFG, as the starting point. It then calculates the weight of each child block of this starting block (i.e., blocks whose parent is the justified block). A block's weight is computed by counting the number of votes it and its descendants have received, considering only the latest vote cast by each validator. Additionally, the block from the most recent slot is given an extra, predefined proposer boost weight. The child block with the highest weight is selected as the new starting point. This process is applied recursively until it reaches a block at the tip of a fork, which is then outputted as the tip of the best chain.

2.2.3.3 The Finality Gadget of Ethereum

The Ethereum protocol employs Casper FFG [26] to checkpoint (finalize) blocks in the best chain. The key idea behind Casper FFG is similar to classical Byzantine Fault Tolerant (BFT)-based consensus protocols. Casper FFG attempts to finalize a checkpoint block, which is the first block of each epoch.

Once a new checkpoint block is finalized, all blocks between it and the previous checkpoint also become finalized. In Casper FFG, for a block to be finalized, it must first receive supermajority approval from at least $\frac{2}{3}$ of the validators to be considered justified. A validator viewing a block as justified implies that, according to that validator's local view, the majority of the network favors the block's finalization. However, this does not guarantee that all validators observe the same approvals in their respective views. To ensure finalization, Casper FFG requires a second round of voting. If a justified block receives a supermajority vote from at least $\frac{2}{3}$ of the validators in this second round, it becomes finalized, ensuring that all honest validators recognize it as a block that received supermajority approval. During each epoch, each validator is assigned to a committee for one slot. In their assigned slot, in addition to casting votes for LMD-GHOST, they also submit two votes for Casper FFG: the source vote and the target vote. The source vote is cast to finalize a previously justified block, while the target vote is used to justify a new block [42, 92].

2.3 Security Analysis of Permissionless Blockchains

The security analysis of a blockchain protocol relies on assumptions about the synchronicity of the underlying network and participant behavior in adhering to the protocol. Below, we first present the well-established definition of blockchain security, outline various network models, and then discuss the security guarantees provided by different blockchain protocols under these network models.

2.3.1 Secure Blockchain Protocols

A blockchain protocol should achieve *consistency* and *liveness* to be considered a secure protocol. For a chain $\mathtt{C}$, let $\mathtt{C}[:-k]$ denote the chain resulting from trimming the k most recent blocks of $\mathtt{C}$. The terms consistency and liveness are formally defined based on security parameters trimming depth k and confirmation waiting time $T_{\mathtt{conf}}$, as follows [93]:

k-consistency: Let $\mathtt{Chain}_i^t$ denote the best chain in the view of honest node i at time t, and $\mathtt{Chain}_j^{t'}$ denote the best chain in the view of honest node j at time t', where $t' \geq t$. It must hold that the chain $\mathtt{Chain}_i^t[:-k]$ is a prefix of the chain $\mathtt{Chain}_j^{t'}$ or they are the same chain.

$(k, T_{\mathtt{conf}})$-liveness: If a transaction $\mathtt{tx}$ is received by an awake honest node at time t, then for any honest node j awake at time t', where $t' \geq t + T_{\mathtt{conf}}$, $\mathtt{tx}$ is included in $\mathtt{Chain}_j^{t'}[:-k]$, where $\mathtt{Chain}_j^{t'}$ denotes the best chain in the view of honest node j at time t'.

Intuitively, the trimming depth represents the number of the most recent blocks that must be removed from the best chain to access the consistent (finalized) chain across all blockchain nodes. The confirmation time refers to the duration required for a transaction to become part of the consistent (finalized) chain.

2.3.2 Network Models

Analyzing the security of distributed systems, including blockchain protocols, relies on assumptions about the synchronicity of the underlying network. In any large-scale distributed system, challenges such as network partitioning or node failures are expected. The distributed systems literature [37, 45] defines three primary network models:

Synchronous network: There is a finite and known upper bound, Δ, such that any message sent at time t is guaranteed to be delivered by $t + \Delta$.

Asynchronous network: Message delivery can be delayed by any finite amount of time, with no upper bound on delivery time.

Partially synchronous network: There exists a finite but unknown Global Stabilization Time (GST) and a finite, known upper bound Δ, such that any message sent at time t is guaranteed to be delivered by $\max(t, \mathtt{GST}) + \Delta$.

Informally, the partially synchronous model assumes the system behaves asynchronously until GST and synchronously thereafter. We focus on analyzing blockchain security under the synchronous and partially synchronous models, as network partitioning and delays, though possible, are typically transient, and we can assume the network will recover over time.

2.3.3 The CAP Theorem

The CAP theorem refers to the trade-offs between **C**onsistency, **A**vailability (liveness), and **P**artition tolerance in distributed systems [50]. According to the blockchain CAP theorem, there is no blockchain protocol that can achieve both liveness under dynamic participation and consistency under network partitions (partially synchronous setting) [68, 77]. This theorem arises from the fact that if a protocol does not hear anything from a node, it cannot distinguish whether the node is offline or the network is partitioned. Therefore, in the context of the CAP theorem, blockchain protocols are generally classified into two categories: live protocols, which prioritize availability, and safe protocols, which prioritize consistency.

2.3.4 Security Analysis of Longest Chain-Based Protocols: Bitcoin and Cardano

Synchronous setting. The Bitcoin blockchain is proven to be secure, achieving both consistency and liveness under synchrony with dynamic participation, provided that the share of awake honest mining power is sufficiently greater than the adversarial mining power [36, 46–48, 83]. Similarly, Ouroboros Genesis, the underlying consensus mechanism of Cardano in future updates, is proven secure in a synchronous and dynamically available setting under the assumption that a majority of awake stakeholders remains honest [17]. It is commonly assumed that longest-chain-based protocols are secure under synchrony as long as more than half of the resources (mining or stake shares) are honest. However, this assumption overlooks the impact of network delay, which can cause some honest blocks to be excluded from the honest chain, thereby reducing the effective share of honest participants (i.e., the fraction of their share that actually contributes to extending the canonical chain).

Partially synchronous setting. Considering the CAP theorem classification, the longest-chain protocol falls into the group of protocols that prioritize liveness under dynamic participation over safety under network partition. In such protocols, when a node becomes unresponsive, others optimistically assume it is offline and continue extending the longest chain. This optimistic design of longest-chain comes at the cost of potential consistency violations if the network is partitioned. Therefore, both the Bitcoin and Cardano protocols are not secure in a partially synchronous network model. No matter how large the trimming parameter k is chosen in the consistency definition, an adversary can select the `GST` (the duration of the asynchrony period) such that, even after trimming k blocks from the honest nodes' longest chains, the resulting chains remain inconsistent.

2.3.5 Security Analysis of Ebb-and-Flow Protocols: Ethereum

The authors in [77] explored the concept of flexible consensus by introducing the ebb-and-flow protocols. These protocols maintain two types of ledgers: an available ledger and a finalized ledger, with the finalized ledger always being a prefix of the available one. The available ledger is required to remain live and consistent under synchronous conditions with dynamic participation, while the finalized ledger must remain consistent even in the presence of network partitions. An aggressive client, prioritizing transaction speed over security, may choose to trust the available ledger. In contrast, a conservative client will rely solely on the finalized ledger, as transactions recorded there are guaranteed not to be reverted.

The Ethereum protocol follows the ebb-and-flow property by using the fork choice rule LMD-GHOST to output an available ledger and, simultaneously, the finality gadget Casper FFG to output a finalized ledger [27].

Synchronous setting. Ebb-and-flow protocols, when equipped with a suitable dynamically available fork choice rule, such as the longest-chain rule, can be proven secure under synchrony with dynamic participation, while tolerating an adversarial resource share of up to 50%. However, to the best of our knowledge, no rigorous security proof has yet been established for the LMD-GHOST protocol. The initial version of LMD-GHOST was shown to be vulnerable to the *balancing attack*, in which an adversary attempts to evenly distribute honest votes across two conflicting forks [77, 92]. To address this vulnerability, Ethereum's implementation of LMD-GHOST introduced the *proposer boost* [25], which assigns additional weight to the block proposed in the most recent slot. While this modification helps mitigate balancing attacks, the additional weight from the proposer boost also increases the risk of *reorganization attacks* in settings with dynamic participation [39]. In summary, the lack of a formal security proof for LMD-GHOST complicates any precise assessment of the Ethereum available ledger's security, especially without relying on the guarantees provided by the finalized ledger.

Partially synchronous setting. The Casper FFG protocol is a BFT-style protocol that, despite losing liveness under dynamic participation, ensures the consistency of Ethereum's finalized ledger even in the presence of network partitions, as long as the adversarial resource share remains below one-third. The Ethereum protocol is designed to provide accountability; that is, if a safety violation occurs, at least one-third of the stake can be provably identified as misbehaving and consequently slashed [10, 26].

2.4 Incentivization Mechanism in Permissionless Blockchains

As discussed in Section 2.3, the security of blockchain protocols fundamentally depends on the assumption that a majority or supermajority of participants behave honestly. This implies that if a significant portion of nodes act maliciously, the security of the blockchain could be compromised. This leads to a critical question: Why should we trust the assumption that the majority of participants will behave as expected?

In permissioned blockchains, this assumption is more reasonable. Nodes are typically operated by a central authority or a consortium of known entities, who have reputational and legal incentives to follow the protocol correctly. These external pressures make it easier to enforce honest behavior. However,

permissionless blockchains, which allow anyone to join and participate, lack inherent accountability. In this context, incentive mechanisms play a crucial role. To promote honest behavior in a decentralized environment, permissionless blockchains must align participants' self-interest with the goals of the protocol. This is typically achieved through economic incentives, such as rewards (e.g., block rewards and transaction fees) and penalties (e.g., slashing), to encourage honest participation and deter malicious actions.

The incentivization mechanism plays a crucial role in shaping participant behavior, indicating that while achieving consistency and liveness is necessary for any blockchain, it is not sufficient to guarantee security in a permissionless setting with decentralized, incentive-driven actors. A thorough security analysis must also consider how the incentive structure influences participant strategies. If a strategy enables a rational, reward-maximizing actor to earn more than their fair share, it may encourage deviation from honest behavior. A fundamental property of any secure permissionless blockchain is *fairness*. A blockchain protocol is considered fair if, for any honest participant, the expected long-term reward share closely matches their active resource share, with only a negligible margin of error [56, 93]. Since block production is a primary source of rewards in blockchain protocols, a key aspect of fairness is how fairly blocks are distributed among participants, referred to as block fairness. This concept is formally defined as follows [80]:

(l, δ)-block fairness: A blockchain protocol satisfies (l, δ)-block fairness if, for any honest participant controlling an α fraction of the total resource share, it holds that, except with negligible probability over the protocol's execution, the fraction of blocks contributed by that participant to the canonical chain over any l consecutive blocks is at least $(1 - \delta)\alpha$.

Intuitively, (l, δ)-block fairness implies that the block ratio of an adversarial participant with resource share $\alpha_{\mathcal{A}}$ is upper bounded by $\alpha_{\mathcal{A}} + \delta(1 - \alpha_{\mathcal{A}})$ over any window of l canonical blocks. Our focus is mainly on long-term fairness (i.e., as $l \to \infty$), since the probabilistic nature of mining or block proposer selection can cause short-term deviations in rewards from a participant's proportional resource contribution. Moreover, we are interested in a high degree of fairness (i.e., as $\delta \to \epsilon$), because even small profits obtained through malicious strategies can incentivize deviation from the protocol. In the remainder of this section, we refer to a protocol as *fair* if it achieves (∞, ϵ)-block fairness. In the following, we analyze the incentivization mechanisms of prominent permissionless blockchains and evaluate their fairness guarantees.

2.4.1 Analysis of the Incentivization Mechanism in Bitcoin

2.4.1.1 Bitcoin Incentives

The incentives in Bitcoin come from two sources [74]:

1. **Protocol reward:** When a miner successfully mines a new block, they receive a specified number of newly minted coins through the coinbase transaction—the first transaction in the block—known as the protocol reward.
2. **Transaction fees:** Miners also collect transaction fees from the transactions they include in the block.

To control inflation, introduce scarcity, and help preserve the value of the cryptocurrency, Bitcoin has a capped total supply [5]. To support this design, the protocol includes a *halving mechanism* that reduces the protocol reward by half every four years.

2.4.1.2 Fairness Analysis of Bitcoin

Bitcoin's longest chain protocol, while achieving liveness, is not a fair protocol. Liveness is a weaker property compared to fairness, as achieving liveness only requires the existence of a single honest block in the chain that includes the transactions [93]. However, to achieve fairness, the number of blocks contributed by each honest node should be proportional to its mining share. There are several attacks that can threaten Bitcoin's fairness, with *selfish mining* [43] being one of the most well-known.

Selfish mining. In the selfish mining attack, an adversarial miner with a mining share below one-half attempts to mine a private chain on top of the public longest chain. This strategy creates a temporary fork in the network, where one branch remains private to the adversary while the other is publicly visible to honest miners. If the adversarial miner is fortunate and mines a private fork that becomes longer than the public one, the adversary can publish this fork, causing honest miners to abandon the shorter non-adversarial chain and switch to the adversary's fork. Consequently, all blocks in the non-adversarial fork that were mined after the fork divergence are discarded and excluded from the best chain.

We will provide a more detailed explanation of selfish mining in Section 2.5. As a result of selfish mining, some honest blocks are removed from the best chain, which in turn increases the adversary's block share. This increase allows the adversary to receive a reward share greater than its fair share, thereby exploiting the fairness of the Bitcoin protocol.

2.4.2 Analysis of the Incentivization Mechanism in Cardano

2.4.2.1 Cardano Incentives

In contrast to Bitcoin, where block rewards are transferred immediately to the miner upon successful block creation, Cardano distributes rewards at the end of each epoch among all validators who participated in that epoch. An epoch in Cardano lasts five days and consists of 432 000 slots.

Rewards in Cardano come from two primary sources [7, 15]:

1. **Monetary expansion:** The Cardano protocol maintains a reserve pot containing a capped number of coins. At the end of each epoch, a small percentage of the remaining reserve is added to the reward pot as part of the monetary expansion. This expansion rate is predefined by the protocol.
2. **Transaction fees:** All transaction fees collected from blocks proposed during the epoch are also added to the epoch's reward pot.

At the end of the epoch, 20% of the accumulated reward pot is allocated to the treasury, while the remaining 80% is distributed among the staking pools as staking rewards. These rewards are allocated proportionally based on each pool's deposited stake share.

However, staking pools do not receive their full staking rewards. The actual reward is adjusted based on the pool's performance, defined as the pool's actual block production ratio during the epoch divided by its expected block production ratio, as determined by its stake share. Additional factors, such as pool saturation, may also influence the final reward distribution.

2.4.2.2 Fairness Analysis of Cardano

The selfish mining attack can threaten the fairness of any longest chain-based blockchains provided that the incentive distribution is correlated to the block ratio. In the initial version of Ouroboros [62], the underlying protocol of Cardano, the authors claimed that the proposed protocol is resistant to selfish mining. This was because, under the reward mechanism they proposed, the epoch reward is distributed based on a pool's stake share, regardless of its actual block ratio during the epoch. Although this mechanism can address the selfish mining issue by making rewards independent of block contribution, it introduces a more significant problem concerning the protocol's fairness. An offline proposer can receive a larger reward than an active proposer simply by holding a greater deposited stake, which is completely unfair.

To improve fairness, the implemented version of Cardano adjusts the reward share based on pool performance, defined as the block ratio divided by the stake

ratio [7]. This implies that a pool's block ratio during the epoch influences its final reward, suggesting that selfish mining-style attacks could also pose a threat to Cardano's fairness.

2.4.3 Analysis of the Incentivization Mechanism in Ethereum

2.4.3.1 Ethereum Incentives

Validator rewards in Ethereum originate from two main sources:

1. **Consensus layer reward:** This includes the incentives paid by the protocol to validators for their participation in the consensus process. According to the Altair upgrade [1] of the Ethereum protocol, a validator's consensus layer reward comprises three components [41]: attestation (vote) rewards, proposer rewards, and sync committee rewards. The attestation reward refers to the reward validators receive for submitting votes to help determine the best available and finalized ledgers. The proposer reward is granted to the block proposer for including attestations (votes) in its proposed block. The sync committee reward is given to validators selected as part of the sync committee, a randomly chosen group of validators selected every 256 epochs who are responsible for signing block headers to assist light clients in tracking the best chain.
2. **Execution layer reward:** This refers to the total reward received by a block proposer for including transactions in a block. If the proposer constructs the transaction bundle themselves, this reward consists of the sum of all priority fees contained in the block. Alternatively, if the proposer delegates bundle creation to a specialized block builder via the proposer-builder separation (PBS) model, the reward is the amount the block builder pays in exchange for access to the proposer's block space [55].

The attestation rewards represent the largest portion (84.4%) of the total consensus layer rewards [41]. Each attestation in Ethereum includes three votes: the head vote (used by the LMD-GHOST protocol), as well as the source and target checkpoint votes (used by the Casper FFG protocol). To be eligible for attestation rewards, votes must be both timely and correct. A vote is considered timely if it is published within the time window specified by the protocol, and correct if it aligns with the chain that ultimately becomes canonical. For more information on Ethereum's consensus rewards, refer to Section 10.A1.1.

2.4.3.2 Fairness Analysis of Ethereum

The Ethereum blockchain, which uses the LMD-GHOST fork-choice rule, is vulnerable to reorganization (reorg) attacks that can exploit its fair reward

distribution mechanism [79, 92]. In a reorganization attack, an adversarial validator attempts to exclude blocks proposed by honest validators from the canonical chain by building an alternative chain on top of older blocks rather than extending the current chain tip. This strategy can be executed by deliberately delaying the publication of the adversary's own blocks and the votes under its control, allowing the adversarial fork to eventually outweigh the publicly visible chain.

A successful reorganization attack can increase the adversarial validator's proposer rewards, i.e., the attestation inclusion rewards. Reorganizing honest blocks causes the attestations included in those blocks to be excluded from the canonical Ethereum chain, allowing the adversarial validator to re-include them in its own blocks and earn the associated rewards. In addition to increasing proposer rewards, the adversary can also boost its execution layer rewards by extracting valuable transactions from the reorged blocks and including them in its own blocks.

Readers are referred to Section 10.A2 for a detailed quantification of reorganization attacks in Ethereum. Preventing such reorgs is a key goal of Ethereum's consensus development, leading to the proposal of new reorg-resilient fork-choice rules to replace LMD-GHOST, such as Goldfish [39] and RLMD-GHOST [32].

2.5 Selfish Mining: A Fairness Attack on Bitcoin

In the selfish mining attack, introduced by Eyal et al. [43], an adversary can achieve a block ratio greater than its mining share, provided that its share is sufficiently large. The core idea behind selfish mining is that the adversary deliberately creates fork races by strategically withholding its blocks. Instead of immediately broadcasting a newly mined block to the network, the adversary withholds it to create a private adversarial fork. This behavior triggers a fork race between a public fork and the adversarial private fork, where only one can ultimately be included in the longest chain. If the adversary wins the fork race, the non-adversarial blocks in the public fork are excluded from the longest chain, resulting in an increase in the adversary's block ratio. The valid Bitcoin blocks that remain outside the longest chain are referred to as orphaned blocks.

2.5.1 Simple Selfish Mining Strategy in Bitcoin

In this section, we describe a simple selfish mining strategy $\pi^{\mathtt{SM}}$ introduced in [43], which applies to a system comprising a group of honest nodes, denoted by $\mathcal{H}$, and an adversary controlling a set of mining nodes, denoted by $\mathcal{A}$. Assume

that all honest mining nodes $\mathcal{H}$ mine on top of a single public chain, while the adversary $\mathcal{A}$ conducts selfish mining and mines on top of an adversarial private chain. These two chains share a common subchain, after which they diverge. The common subchain is referred to as the *canonical* chain, whose included blocks cannot later be orphaned by the adversarial chain. The adversary's chain (honest chain) proposed on top of the canonical chain tip is referred to as the adversarial fork (honest fork). Let $l_{\mathcal{A}}$ ($l_{\mathcal{H}}$) denote the length of the adversarial fork (honest fork). We denote by $S_{l_{\mathcal{A}},l_{\mathcal{H}}}$ the state of the fork race. The simple selfish mining strategy $\pi^{\mathtt{SM}}$ is as follows:

- The initial state of the fork race is $S_{0,0}$, where both the adversary and honest miners mine on top of the same canonical chain.
- If the system is in state $S_{0,0}$ and the next block is mined by the honest nodes, its miner publishes the block immediately to the network. In this case, all the nodes, including the adversary, adopt the new block as the tip of the canonical chain. The system remains in the same state $S_{0,0}$.
- If the system is in state $S_{0,0}$ and the adversary mines the next block, it withholds the block. The adversarial fork achieves a one-block lead over the honest fork, and the system transitions to $S_{1,0}$.
- If the system is in state $S_{1,0}$ and the next block is mined by the adversary, the adversarial fork achieves a two-block lead over the honest fork, and the system transitions to $S_{2,0}$. The adversary keeps its adversarial fork private until the honest fork reduces the lead to one block, at which point it publishes its fork. This strategy guarantees that, at the publishing time of the adversarial fork, the condition $l_{\mathcal{A}} > l_{\mathcal{H}}$ holds, implying that the adversarial fork is the longest fork in the network. Therefore, all the nodes, including the honest nodes, adopt the adversarial fork as the longest chain in their view, and the system transitions to the initial state $S_{0,0}$. Note that under this scenario, all the blocks in the honest fork become orphaned.
- If the system is in state $S_{1,0}$ and the next block is mined by the honest nodes, the adversary immediately publishes its private block. As a result, a fork race occurs between two forks of the same height of one block. Under a same-height fork race, the adversary mines on top of the adversarial fork, while the honest nodes mine on top of the fork they first observe in their view. The probability of the adversary winning the same-height fork race depends on how quickly the adversary can propagate its blocks through the network. If the next block is mined on top of the adversarial (honest) fork, the honest (adversarial) block in the competing fork becomes orphaned, and the state transitions to the initial state $S_{0,0}$.

2.5.2 Optimal Selfish Mining Strategy in Bitcoin

Despite the introduction of selfish mining in [43], it turned out that the introduced strategy is not the optimal strategy to maximize the block ratio. In [85], the authors implemented a Markov Decision Process (MDP)-based tool to determine the optimal selfish mining strategy $\pi^{\texttt{OSM}}$. MDP is a mathematical tool used for decision-making, where an agent makes decisions in a stochastic environment to maximize its cumulative rewards. MDP consists of states, actions, transition probabilities, and rewards, enabling the derivation of an optimal policy, which specifies the best action to take in each state [94].

To model the selfish mining states in [85], the authors have used three parameters: $l_{\mathcal{A}}$ denoting the adversarial fork length, $l_{\mathcal{H}}$ denoting the honest fork length, and `fork` storing information about the miner of the latest block and its publication status. In [85], 4 different actions have been considered for the adversary:

- `adopt`: Adopt the honest fork as the longest fork.
- `override`: Publish an adversarial fork that is one block longer than the honest fork to override it.
- `wait`: Continue mining on the private fork without publishing or adopting.
- `match`: Publish an adversarial fork matching the height of the honest fork to initiate a same-hight fork race.

The optimal selfish mining strategy $\pi^{\texttt{OSM}}$ obtained using the MDP analysis in [85] can dominate the simple selfish mining strategy $\pi^{\texttt{SM}}$ introduced in [43] in achieving a higher block ratio. This increase in block ratio arises as the best action prescribed by strategy $\pi^{\texttt{OSM}}$ at specific states differs from that of $\pi^{\texttt{SM}}$. For instance, if the adversary's mining share is sufficiently large, the best action to take when the adversarial fork is one-block behind is not to adopt the honest fork as suggested in $\pi^{\texttt{SM}}$, but is to continue mining on top of the adversarial fork in hope of compensating the lag and winning the fork race. As another example, when the adversarial fork has a lead of two or more blocks, the optimal strategy $\pi^{\texttt{OSM}}$ frequently uses the `match` action to initiate same-height fork races. By winning these races, the adversary can secure blocks deeper in the adversarial fork while keeping its recent blocks private to initiate additional fork races.

2.5.3 Mining Share Threshold for Profitable Selfish Mining

According to analysis conducted in papers [43, 85], selfish mining can dominate honest mining only if the adversary controls a sufficiently large portion of the mining power. The mining power threshold for a profitable selfish mining attack depends on the adversary's communication capability, as during same-height fork races, the honest nodes choose to mine on top of a fork that they have received first in their view. Therefore, if the adversarial nodes are well-distributed and

well-connected to the honest nodes, they can propagate the adversarial fork faster than the honest fork through the network, thus increasing their chance of winning the fork race. The normal adversarial communication capability can be assumed to be 50%, implying that during a same-height fork race, half of the honest nodes receive the adversarial fork first. It is shown that under the normal communication capability of 50%, the mining share threshold for profitable selfish mining is equal to 25% [43, 85]. This threshold, although requiring a considerable amount of mining power, is still in the range of the largest Bitcoin mining pool's share [13].

2.5.4 Background on the Time-Averaged Profitability of Selfish Mining

Selfish mining can increase the adversarial block ratio by orphaning some of the honest blocks. However, how this increase in block ratio and the exclusion of honest blocks translate into adversarial profit is not immediately obvious. In this section, we address this question by analyzing how the time-averaged profit (i.e., the profit per unit time) achieved by a selfish miner behaves from the beginning of the attack [90]. For the analysis in this section, consider an adversary $\mathcal{A}$ whose mining share is equal to $\alpha_{\mathcal{A}}$. Assume that if $\mathcal{A}$ performs selfish mining, it can achieve a block ratio of $\mathtt{BR}_{\mathcal{A}}^{\mathtt{SM}}$, where $\mathtt{BR}_{\mathcal{A}}^{\mathtt{SM}} > \alpha_{\mathcal{A}}$.

To analyze the time-averaged profit of mining strategies, we introduce the concepts of *total block mining rate* and *canonical block mining rate*. The total block mining rate refers to the rate at which a miner produces blocks, including both orphaned blocks and canonical blocks, whereas the canonical block mining rate refers to the rate at which a miner produces canonical blocks. Let λ denote the total block mining rate of the entire network when all Bitcoin nodes, including the adversary, follow honest mining. Ignoring the natural occurrence of orphans due to network issues, the canonical block mining rate of the entire network under honest mining is also equal to λ.

Note that if mining costs are ignored[3] and the Bitcoin block reward is normalized to 1, the time-averaged profit of a mining strategy is equivalent to the canonical block mining rate achieved under that strategy. Thus, to compare the profitability of selfish and honest mining, we need to compare their respective canonical block mining rates.

[3]Mining costs can be ignored when comparing the profitability of two mining strategies if, in both strategies, the adversary utilizes its entire mining power, as is the case for both honest and selfish mining.

2.5.4.1 Selfish Mining Unprofitability Before Difficulty Adjustment

To explain the unprofitability of selfish mining before difficulty adjustment, we compare the adversary's profitability when following honest mining versus selfish mining.

Honest mining: If the adversary $\mathcal{A}$ follows the honest strategy in $\texttt{epoch}_1$, its total block mining rate equals $\lambda\alpha_{\mathcal{A}}$. Its canonical block mining rate also equals $\lambda\alpha_{\mathcal{A}}$, as there is no ongoing attack and thus no block orphaning.

Selfish mining: Consider that the adversary $\mathcal{A}$ begins the selfish mining attack at the start of $\texttt{epoch}_1$. Since the mining difficulty for $\texttt{epoch}_1$ is predetermined before the epoch starts, the adversary's strategy during $\texttt{epoch}_1$ cannot influence the mining difficulty during this epoch. Therefore, the total block generation rate under selfish mining in $\texttt{epoch}_1$ remains the same as $\lambda\alpha_{\mathcal{A}}$ achieved under honest mining. However, the adversary's canonical block mining rate decreases in $\texttt{epoch}_1$ under selfish mining. This is because selfish mining is a risky attack in which a withheld adversarial block may get orphaned. The orphaning of some adversarial blocks under selfish mining results in the adversary's canonical block generation rate being less than its total block generation rate.

The comparison between scenarios above shows that the canonical block generation rate during the first epoch of selfish mining is less than what can be achieved under honest mining. Note that selfish mining may increase the adversary's block ratio during the first epoch of the attack. However, despite this increase in block ratio, the adversary cannot gain a higher time-averaged profit during the first epoch of the attack.

2.5.4.2 The Impact of Selfish Mining on Mining Difficulty

Let $\texttt{epoch}_1$ and $\texttt{epoch}_2$ denote two consecutive epochs. Recall that a Bitcoin epoch is the duration in which $L = 2016$ blocks are added to the canonical chain, ideally taking two weeks. If the adversary $\mathcal{A}$ starts selfish mining in $\texttt{epoch}_1$, some blocks, both honest and adversarial, become orphaned in $\texttt{epoch}_1$. As a result, the duration required to add L blocks to the canonical chain takes longer than the ideal epoch duration of two weeks. The extended duration of $\texttt{epoch}_1$ leads to a decrease in the mining difficulty of the subsequent epoch, $\texttt{epoch}_2$. This is because DAM interprets the prolonged duration of $\texttt{epoch}_1$ as an indication that the mining difficulty was set too high for the available network mining power during the previous epoch and should therefore be reduced.

2.5.4.3 Selfish Mining Profitability After Difficulty Adjustment

Applying DAM at the end of the first epoch of selfish mining reduces mining difficulty, leading to an increase in the total block mining rate. DAM adjusts the difficulty of the second and later epochs under selfish mining in a way that their duration recovers to the ideal duration of two weeks. Therefore, during the second and subsequent epochs, the adversary's canonical block mining rate equals $\lambda \mathtt{BR}_{\mathcal{A}}^{SM}$, which exceeds the canonical block mining rate of $\lambda\alpha_{\mathcal{A}}$ achieved under honest mining. This implies that starting from the second epoch, selfish mining becomes profitable.

In practice, the initial loss period of selfish mining is much longer than one epoch as the adversary's profit in the second epoch, or even later epochs, must first compensate for the losses incurred during the first epoch of selfish mining.

2.5.5 Intermittent Selfish Mining and Profit Lag

To shorten the initial loss period associated with selfish mining, Negy et al. [76] introduced the intermittent selfish mining (ISM) attack, in which the adversary alternates between selfish mining and honest mining at every new epoch. Specifically, for two consecutive epochs, $\mathtt{epoch}_1$ and $\mathtt{epoch}_2$, the adversary follows selfish mining in $\mathtt{epoch}_1$ and reverts to honest mining in $\mathtt{epoch}_2$. As discussed earlier, selfish mining in $\mathtt{epoch}_1$ reduces the mining difficulty at the start of $\mathtt{epoch}_2$. By reverting to honest mining in $\mathtt{epoch}_2$, the adversary can fully exploit the reduced mining difficulty. Under honest mining, none of the adversarial blocks in $\mathtt{epoch}_2$ are orphaned, allowing the adversary to maximize its canonical block mining rate, i.e., time-averaged profit, during this epoch. Negy et al. [76] concluded that the increased profit achieved during $\mathtt{epoch}_2$ through intermittent selfish mining contributes to reducing the initial loss period.

Grunspan and Pérez-Marco [54] revisited the conclusion of [76], arguing that while ISM may outperform honest mining by the end of the second epoch, its profitability drops below that of honest mining once the third epoch begins and the adversary returns to selfish mining. They introduced the concept of *profit lag* to better assess the initial loss period of a mining strategy. The profit lag of a mining strategy π is defined as the earliest time point, since the deployment of strategy π, after which the adversary's time-averaged profit consistently exceeds that of honest mining. Grunspan and Pérez-Marco [54] demonstrated that intermittent selfish mining experiences a considerably longer profit lag compared to continuous selfish mining, indicating that intermittent selfish mining is not an effective strategy for reducing the initial loss period of selfish mining.

Chapter 3

Incentive Analysis in Bitcoin

3.1 Time-Averaged Analysis of Selfish Mining

This section provides an overview of our work in [90], which is presented in Chapter 6. Most of the literature on the selfish mining attack (introduced in Section 2.5) focuses on analyzing the block ratio achieved by the attack, and thus ignoring its temporal aspects [19, 43, 53, 85, 87, 101]. While the block ratio metric provides valuable insights into the long-term profitability of selfish mining, it is not a comprehensive benchmark for evaluating the attack. This is because block ratio analysis typically overlooks the impact of Bitcoin's Difficulty Adjustment Mechanism (DAM) (see Section 2.2.1.3) and, therefore, offers no insight into the attack's profitability before difficulty adjustment occurs.

In a nutshell, even if selfish mining results in a higher block ratio for the adversary, it does not become immediately profitable in terms of the Bitcoin rewards the adversary receives. At least a two-week period (the duration of an ideal Bitcoin epoch) is required for the attack to become profitable, meaning that the attack is always accompanied by an initial period of non-profitability or even profit loss (see Section 2.5.4). This prolonged initial loss could act as a demotivating factor for Bitcoin nodes or mining pools to conduct selfish mining.

Our main contributions in [90] are as follows:

- Introduction of smart *intermittent selfish mining (version 1)*, which achieves a lower profit lag than prior selfish mining strategies (Section 3.1.1).
- Proposal of modified Bitcoin DAMs that account for orphan blocks (Section 3.1.2).
- Introduction of *smart intermittent selfish mining (version 2)* and the *orphan exclusion attack*, both effective even under the modified DAMs (Section 3.1.3).

3.1.1 The Smart Intermittent Selfish Mining (Version 1)

In Section 6.5.2, we introduce the smart intermittent selfish mining (version 1), denoted as SISM1, which features a shorter initial loss period compared to both consistent selfish mining [43, 85] and the intermittent selfish mining (ISM) attack [76] (see Section 2.5.5). Additionally, SISM1 outperforms ISM in long-term profitability. In ISM, the adversary performs selfish mining only during odd epochs, resulting in a ratio of the selfish mining period length, measured by the number of canonical blocks, to the canonical chain length being equal to 0.5. The time-averaged profit of SISM1 exceeds that of ISM while maintaining the same ratio of selfish mining period length to canonical chain length at 0.5.

3.1.1.1 SISM1 Description

Consider two consecutive epochs, denoted as $\texttt{epoch}_{\texttt{odd}}$ and $\texttt{epoch}_{\texttt{even}}$, during which a total of $2L$ blocks are added to the canonical chain. In the SISM1 strategy, the adversary engages in selfish mining for $(1-\eta)L$ blocks during $\texttt{epoch}_{\texttt{odd}}$ and for ηL blocks during $\texttt{epoch}_{\texttt{even}}$, where η lies in the range $[0, 0.5]$. For the rest of the blocks in both epochs, the adversary adheres to the honest mining protocol. By design, SISM1 ensures that the ratio of the selfish mining period length to the total canonical chain length is 0.5. Note that when $\eta = 0$, SISM1 is equivalent to the standard intermittent selfish mining attack.

3.1.1.2 SISM1 Profitability Analysis

Our analysis shows that the time-averaged profit of SISM1 strategy maximizes when $\eta = 0.5$, that is, when the adversary performs selfish mining for half of the blocks in each epoch. By engaging in selfish mining for exactly half of the blocks in each epoch, the mining difficulty for the second and subsequent epochs remains stable. This suggests that, to maximize profit, the adversary should avoid introducing fluctuations in the mining difficulty and maintain it at a consistent level.

In Section 6.5.1, we show that the mining share threshold for a profitable ISM attack is 0.2773, which is higher than the 0.25 threshold for the consistent selfish mining attack. The minimum mining share needed for SISM1 to succeed is 0.25, thus recovering the threshold of consistent selfish mining and allowing attacks with a smaller share. Furthermore, in Section 6.5.2, we demonstrate that the proposed SISM1 strategy has a shorter initial loss period compared to both consistent selfish mining and ISM, indicating that an adversary seeking to achieve profitability in the shortest time should adopt the SISM1 strategy. For

example, for an adversary with a mining power share of $\frac{1}{3}$ and a communication capability of 0.5, the profit lags for ISM and consistent selfish mining are approximately 24 and 6 weeks, respectively. In contrast, the profit lag for the same adversary using SISM1 is reduced to approximately 4 weeks (see Figure 6.3).

3.1.2 Modified Difficulty Adjustment Mechanism

In [90], we formally analyze the claim made in [53] that incorporating the count of orphan blocks into the DAM calculation can make selfish mining completely unprofitable. We demonstrate that, although this countermeasure can limit the time-averaged profit to an acceptable level, there still exist adversarial strategies capable of outperforming honest mining, even when orphans are incorporated into DAM.

3.1.2.1 Orphan Block Reporting

The primary reason behind the profitability of selfish mining is the current Bitcoin DAM, which reduces mining difficulty when the blockchain is subjected to a selfish mining attack [53, 76] (see Section 2.5.4). Specifically, the Bitcoin DAM cannot distinguish whether an extended epoch duration is due to some mining power being offline or the chain being under a selfish mining attack. This is because the Bitcoin DAM only tracks the count of canonical blocks to estimate the available hash power. The authors in [53] proposed enhancing the DAM by incorporating the number of orphan blocks, in addition to canonical blocks. Orphan blocks serve as an indication that the blockchain is under attack and some mining power is being wasted. By considering orphan blocks, the Bitcoin DAM can better estimate active mining power, which would prevent the reduction in mining difficulty during a selfish mining attack. As long as mining difficulty is not reduced, honest mining remains the most profitable strategy. To this end, the authors in [53] concluded that incorporating the number of orphan blocks into the DAM could make selfish mining completely unprofitable.

In [90], we propose two modified versions of Bitcoin's current DAM. These modified DAMs represent the most straightforward approaches to extending the Bitcoin DAM to account for orphan blocks. The proposed modified DAMs differ from each other in how the difficulty epoch is defined. Recall that, according to Bitcoin's current DAM, an epoch is the period during which $L = 2016$ blocks are added to the canonical chain.

3.1.2.2 Modified DAM with a Fixed Total Block Count per Epoch

In Section 6.6.2, we introduce a version of modified DAM denoted by $\mathsf{DAM}_1^{\texttt{modified}}$. An epoch in $\mathsf{DAM}_1^{\texttt{modified}}$ is defined as the duration in which a set of L blocks, comprising both canonical and orphan blocks, are mined and subsequently reported in the canonical chain. Therefore, under $\mathsf{DAM}_1^{\texttt{modified}}$, the total number of blocks mined per epoch is fixed; however, the number of canonical blocks can vary across different epochs.

To consider orphan block counts in the DAM, orphan blocks must be reported on the canonical chain. This can be achieved by honest mining nodes including the headers of orphan blocks they observe in their blocks. Since a block header is significantly smaller than the full block[1], reporting orphan block headers can be considered a highly efficient countermeasure against selfish mining.

3.1.2.3 Modified DAM with a Fixed Canonical Block Count per Epoch

In Section 6.6.3, we present another modified version of DAM, denoted by $\mathsf{DAM}_2^{\texttt{modified}}$. An epoch in $\mathsf{DAM}_2^{\texttt{modified}}$ is defined as the duration during which a set of L canonical blocks are mined. Under $\mathsf{DAM}_2^{\texttt{modified}}$, the number of canonical blocks mined per epoch is fixed; however, the total number of blocks mined per epoch can vary across different epochs.

3.1.3 Potential Attacks on Modified DAMs

In [90], we introduce two attacks: *smart intermittent selfish mining (version 2)* and the *orphan exclusion attack*, both of which can threaten Bitcoin under the modified DAMs. The existence of these attacks suggests that modifying Bitcoin's DAM to account for the count of orphan blocks is insufficient to completely eliminate the profitability of selfish mining.

3.1.3.1 Smart Intermittent Selfish Mining (Version 2)

Section 6.7 presents the smart intermittent selfish mining (version 2) attack, abbreviated as SISM2. In this attack, during odd epochs, the adversary divides its mining power share $\alpha_{\mathcal{A}}$ into two parts: idle mining power $e\alpha_{\mathcal{A}}$ and honest mining power $(1-e)\alpha_{\mathcal{A}}$, where $0 \leq e \leq 1$. During even epochs, however, the adversary uses its entire mining power share to perform selfish mining.

[1]The Bitcoin block header is 80 bytes, while a typical Bitcoin block size exceeds 1.5 megabytes [2].

Our analysis shows that SISM2 can be profitable under both $\mathsf{DAM}_1^{\texttt{modified}}$ and $\mathsf{DAM}_2^{\texttt{modified}}$; however, its profitability is higher under $\mathsf{DAM}_2^{\texttt{modified}}$ (see Figures 6.4 and 6.5). SISM2 is particularly advantageous for miners with a relatively low profitability factor, defined as the return per unit of money invested by the miner [72]. Note that although selfish mining during the even epochs of SISM2 can be detected under the modified DAMs, intentionally leaving a portion of the adversary's mining power idle during the odd epochs can still reduce the mining difficulty, thereby contributing to the profitability of SISM2.

3.1.3.2 Orphan Exclusion Attack

In Section 6.8, we introduce the orphan exclusion attack. The goal of the adversary in this attack is to orphan a few honest blocks in a way that the canonical honest blocks cannot report them. This lack of reporting excludes these orphaned blocks from the DAM calculation, thereby reducing the mining difficulty in favor of the adversary.

In the orphan exclusion attack, as depicted in Figure 6.6, the adversary attempts to orphan a sequence of consecutive honest blocks at the end of each epoch, including the final honest block of the epoch. As the epoch nears its end, the adversary creates a fork race by mining a private chain. The attack is considered successful if two conditions are satisfied: first, the adversary's private fork succeeds in orphaning the public fork; and second, the final block of the epoch is part of the adversary's private fork. The success of the orphan exclusion attack implies that all honest blocks in the public fork are orphaned and, most importantly, remain unreported. This lack of reporting occurs because no honest block is included in the canonical chain between the start of the attack and the end of the epoch where DAM is applied. As a result, DAM remains unaware of these orphaned blocks.

3.1.3.3 Profitability of the Orphan Exclusion Attack

In Section 6.8.2, we use both an implementation and a theoretical analysis to measure the length of the orphan exclusion attack, defined as the number of honest blocks that become orphaned and remain unreported at the end of each epoch, under both versions of the modified DAM. Our analysis shows that the length of the orphan exclusion attack under $\mathsf{DAM}_2^{\texttt{modified}}$ is greater than that under $\mathsf{DAM}_1^{\texttt{modified}}$, suggesting that the attack poses a greater threat under $\mathsf{DAM}_2^{\texttt{modified}}$.

In Section 6.9, we analyze the time-averaged profit of selfish mining when combined with the orphan exclusion attack under the modified DAMs. Our

analysis shows that selfish mining remains profitable even when orphans are incorporated into DAM. This profitability arises from the reduction in mining difficulty caused by unreported honest blocks. This reduction enables the adversary to mine a greater number of blocks per unit of time. Since the length of the orphan exclusion attack under $\mathsf{DAM}_2^{\texttt{modified}}$ is greater than that under $\mathsf{DAM}_1^{\texttt{modified}}$, selfish mining under $\mathsf{DAM}_2^{\texttt{modified}}$ results in higher profit. Furthermore, our analysis reveals that shorter epoch lengths can amplify the destructive impact of the orphan exclusion attack (see Figure 6.8). This finding highlights a trade-off in DAM design between two goals: resistance to the orphan exclusion attack and sensitivity to hash rate fluctuations.

3.1.4 Effectiveness of Modified DAMs Against Selfish Mining

In Section 6.10, we provide a comprehensive discussion of existing selfish mining countermeasures, highlighting their practical limitations. Focusing on modified DAMs, incorporating orphan blocks can effectively discourage payoff-maximizing selfish miners, making the proposed DAMs a viable countermeasure. However, these modified DAMs are not flawless and cannot mitigate all adversarial strategies. A Byzantine adversary, unconcerned with profitability, may engage in selfish mining solely to reduce the block generation rate and undermine honest miners' profitability. Moreover, the modified DAMs cannot detect or counter profitable adversarial strategies that do not produce orphan blocks. Examples include *coin hopping*, where a miner alternates between blockchains using the same PoW mechanism (e.g., Bitcoin and Bitcoin Cash) [54, 57, 65], and *smart mining* [71], where the adversary alternates between epochs of honest mining and idle periods.

3.2 Security Analysis of Bitcoin in a Semi-Rational Setting

This section provides an overview of our work in [88], which is presented in Chapter 7. The security analyses of numerous attacks on the Bitcoin blockchain proposed in the literature often assume a simplified environment in which, apart from an adversarial mining pool, the remaining mining power is either controlled by honest miners who strictly follow the protocol under all circumstances or by infinitesimally small rational nodes that are incapable of coordinating their strategies as a mining pool [19, 20, 43, 53, 85, 87, 101]. However, the assumption that all non-adversarial nodes are completely honest is far from realistic, as most Bitcoin nodes are rational entities that seek to maximize their own incentives. Accounting for rational mining pools in the Bitcoin ecosystem introduces new challenges for the security analysis, as such pools may control a significant portion of the mining power and can strategically respond to ongoing attacks.

The assumption of a rational setting can, in some cases, favor the adversary and strengthen the attacks, while in other scenarios, it may work against the adversary and reduce the attack's effectiveness. For example, in a rational setting, the adversary may employ incentive manipulation attacks [59], such as bribery [21, 58, 70, 71, 73, 96], to persuade other mining pools to adopt the adversary's preferred strategy, thereby increasing the likelihood of a successful attack—this scenario works in favor of the adversary. On the other hand, in a fork race where the adversary publishes a longer fork, a non-adversarial mining pool with a block on the shorter fork may choose not to adopt the adversary's fork, recognizing that mining on the shorter fork and reviving its block would be more profitable—this works against the adversary.

In [88], we extend the Bitcoin security analysis to a more realistic environment by considering the presence of *petty-compliant mining pools*, defined as pools that adopt the fork with the highest expected return rather than the longest chain (see Definitions 7.2.3, 7.2.4, and 7.2.5). Our main contributions in [88] are as follows:

- Selfish mining analysis in the presence of petty-compliant mining pools (Section 3.2.1).
- Proposing a novel bribery attack that aims to waste a portion of mining power, favoring even small mining pools (Section 3.2.2).
- Introducing a novel distraction attack that targets mining power and remains undetectable by DAM (Section 3.2.3).

3.2.1 Selfish Mining in a Semi-Rational Setting

Accounting for petty-compliant mining pools can influence the distribution of mining power supporting either the adversarial fork or its rival fork during a fork race. Figure 7.1 intuitively illustrates the mining share distributions between forks in a fork race, in the altruistic setting, in the presence of infinitesimally small petty-compliant mining nodes, and in the presence of petty-compliant mining pools. In Figure 7.1, the adversarial block, shown in red, is published after the rival block and offers a bribe to any miner who chooses to mine a new block on top of it. The main takeaway from Figure 7.1 is that with petty-compliant pools, a bribe on the adversarial fork does not ensure full adoption, as victim pools in the rival fork may defend their blocks and counter the adversarial behavior.

3.2.1.1 Impact of Residual Centralization Factor on Selfish Mining

We define the *residual centralization factor* with respect to an adversarial mining pool as a metric for evaluating the distribution of mining share among non-adversarial pools (see Definition 7.3.2). The more distributed the non-adversarial mining share, the lower the residual centralization factor for the adversarial pool. Our theoretical analysis in Section 7.3 demonstrates that the residual centralization factor has a strong influence on the profitability of selfish mining in the semi-rational setting. Specifically, Theorem 7.1 states that an adversarial mining pool with a lower residual centralization factor has a higher likelihood of conducting profitable selfish mining. This is because when the non-adversarial mining power is well-distributed (i.e., the residual centralization factor is low), offering a bribe on the adversarial fork can lead to a substantial majority of miners adopting it. In contrast, if large non-adversarial mining pools exist (i.e., the residual centralization factor is high), they act as strong competitors in the fork race, thereby reducing the adversary's chances of success.

3.2.1.2 Profitable Selfish Mining Threshold in Semi-Rational Setting

In an altruistic setting, an important consideration is that an adversary with normal communication capability must control at least 25% of the network's mining power to effectively execute a selfish mining attack [43]. While such thresholds might suggest that selfish mining poses no threat if no single pool exceeds the threshold, our analysis in Section 7.3 highlights that the adversary's mining share is not the sole factor determining the viability of selfish mining in practice. In fact, the distribution of mining power among pools and the gap between their shares are crucial for network security. Even if all mining pools hold less power than the established thresholds, a pool with a significant

share advantage over others may still find selfish mining profitable. In a semi-rational setting with non-adversarial miners holding only infinitesimal shares, the security threshold for profitable selfish mining can reduce to ϵ.

3.2.1.3 Markov Decision Process-Based Analysis of Selfish Mining in Semi-Rational Setting

In Section 7.A3.8, we introduce a Markov Decision Process (MDP)-based tool for determining the optimal selfish mining strategy in a semi-rational environment, where the miner of each block is known. Our MDP framework is specifically designed to account for the presence of petty-compliant mining pools. In the altruistic setting, the only information about the non-adversarial fork that needs to be stored in the state is its length. However, to extend the MDP to a semi-rational setting, the state must also capture the number of blocks owned by different mining pools within the non-adversarial fork, as this information influences the behavior of petty-compliant pools. We present the results of our MDP-based implementation under various mining share distribution settings in Section 7.A3.8. The outcomes of the implementation align with our theoretical analysis, confirming that the more uniformly distributed the non-adversarial mining power is, the greater the reward share obtained by the adversary.

3.2.2 Bribery Attack

The profitability of selfish mining significantly decreases as the mining share of a pool diminishes. This limitation arises primarily because selfish mining is an inherently risky strategy, requiring the adversary to gamble on its own blocks.

In Section 7.4, we introduce a novel *bribery attack* that can dominate both honest mining and selfish mining for small mining pools. Bribery attacks have been previously explored in the context of Bitcoin security, with objectives ranging from censoring a single transaction [73, 96] to executing a successful double-spending attack [21, 58, 70, 71]. Our proposed bribery attack differs from those discussed in the literature, as it aims to destroy a portion of the mining power, similar to the selfish mining attack.

3.2.2.1 Overview of the Bribery Attack

In a nutshell, in the proposed bribery attack, an adversarial mining pool pays a bribe to incentivize petty-compliant mining pools to orphan a block mined by a victim mining pool. The profitability of the bribery attack follows the same principle as selfish mining, as both attacks aim to destroy part of the hash power to exploit the DAM, thereby reducing mining difficulty. The key

difference is that in selfish mining, the adversarial pool uses its own blocks to carry out the attack, whereas in the bribery attack, it uses its budget to incentivize other mining pools to conduct the attack. The superiority of the bribery attack over selfish mining for small pools can be attributed to two factors. First, the bribery attack can be implemented in a way where the bribe is only payable upon successfully orphaning a block, making it risk-free for the adversary. Second, unlike selfish mining, the bribery attack is not limited to times when the adversary has mined a block, allowing it to be conducted more frequently. Section 7.4.1 presents a detailed explanation of the bribery attack implemented via smart contract in the setting where miners are known.

3.2.2.2 Quantifying the Bribery Attack

A key question in the context of the bribery attack is determining the maximum bribe the adversary can offer while still maintaining profitability. To address this, we present Theorem 7.2, which states that the bribery attack can outperform honest mining as long as the normalized bribe amount (see Definition 7.2.8), payable upon successfully orphaning a target block, is less than the adversarial mining pool's share.

In Section 7.A4.5, we use a Markov chain model to analyze the profitability of the bribery attack. Figure 7.3a illustrates the percentage increase in reward share for the eight major mining pools based on the distribution in Table 7.1 under three strategies: the smart contract-based bribery attack, the selfish mining attack, and the undercutting attack [28]. Our results show that the bribery attack is not only profitable even for small mining pools (with a share significantly less than 25%), but also that it outperforms both selfish mining and undercutting attacks for these smaller pools.

3.2.3 Distraction Attack

Both selfish mining and the proposed bribery attack result in the creation of orphan blocks, which can be considered valid proof of mining power destruction attacks. As discussed earlier, these attacks that generate orphan blocks can be mitigated and made unprofitable by modifying the Bitcoin DAM to account for the orphan blocks. In Section 7.A5, we introduce a novel *distraction attack*, which destroys a portion of mining power by diverting it to mine on another blockchain platform. Unlike selfish mining or the proposed bribery attack, the distraction attack does not produce orphan blocks, making it more challenging to mitigate without relying on external trusted platforms.

3.2.3.1 Overview of the Distraction Attack

The main idea of this attack is as follows: after the adversarial mining pool successfully mines a block under the Bitcoin difficulty denoted by $\texttt{Diff}_1$, it withholds the block from the rest of the mining pools and communicates the following statement to them: "The next block in the canonical chain is already mined. To view the content of this block, you must provide a valid *mini-PoW* with a difficulty of $\texttt{Diff}_2$, where $\texttt{Diff}_2 < \texttt{Diff}_1$." This attack creates a strategic dilemma for the petty-compliant mining pools: if they continue mining on the Bitcoin canonical chain and successfully mine the next block, they risk initiating a fork race with the adversarial block, where their block may end up being orphaned. Alternatively, if they choose to access the adversarial block, they must divert some of their mining power to mine a mini-PoW.

In Section 7.A5, we demonstrate that by carefully selecting an appropriate value for $\texttt{Diff}_2$, the adversarial mining pool can incentivize petty-compliant mining pools to prioritize solving the external puzzle, thereby diverting a portion of their resources to a task outside the Bitcoin protocol.

3.2.3.2 Quantifying the Distraction Attack

We present Lemma 7.11 to determine the maximum normalized bribe that the adversary can offer upon successfully mining a mini-PoW, while still keeping the distraction attack profitable. In Section 7.A5.1, we compute the expected returns of both mini-PoW mining and Bitcoin mining under a distraction attack, and demonstrate that non-adversarial miners can be incentivized to mine mini-PoWs instead of Bitcoin blocks. Our analysis reveals that a lower difficulty ratio of $\frac{\texttt{Diff}_1}{\texttt{Diff}_2}$ increases the profitability of the distraction attack for the adversary. However, setting the difficulty ratio too low decreases the number of petty-compliant miners incentivized to participate in mini-PoW mining.

3.3 Bitcoin under Volatile Block Rewards

This section provides an overview of our work in [86], which is presented in Chapter 8. Bitcoin has two main sources of reward: the transaction fee and the protocol reward. The Bitcoin protocol is equipped with a halving mechanism that reduces the protocol reward by half every four years. In the early years of Bitcoin, transaction fees were negligible compared to the high protocol rewards, making the total block reward nearly equal to the fixed protocol reward across all blocks. As a result, the assumption of a fixed block reward for analyzing Bitcoin's security was reasonable during this period. However, as more halvings have occurred and the protocol reward has diminished, the contribution of transaction fees to block rewards has grown significantly. Over time, variations in transaction congestion within the network may cause transaction fees to fluctuate considerably, introducing volatility to Bitcoin's block rewards. This volatility can alter the incentive dynamics of the Bitcoin network, potentially giving rise to new security threats or intensifying existing ones.

In [86], we analyze Bitcoin security under a volatile block reward model. This model is relevant in current days when transaction fees are high and comparable to the protocol reward, and also in the near future, where transaction fees are expected to become the primary source of reward. Some research has been conducted in the literature to analyze Bitcoin security under the transaction-fee setting [19, 28]; however, these studies rely on highly simplified volatile models that overlook the complexities of Bitcoin's mempool behavior. This section enhances the analysis of mining strategies in Bitcoin by incorporating a more realistic volatile block reward model that closely mimics the actual Bitcoin's mempool. The contributions of this section of the thesis are outlined below:

- Revisiting the security threshold of Bitcoin mining in the transaction-fee setting presented in [19] (Section 3.3.1).
- Introducing an MDP-based tool to analyze Bitcoin security under a model referred to as the *simplified volatile reward model* (Section 3.3.2).
- Implementing a reinforcement learning tool using the Asynchronous Advantage Actor-Critic (A3C) algorithm to derive a near-optimal mining strategy under realistic mempool implementations (Section 3.3.3).

3.3.1 Revisiting the Bitcoin Security Threshold

Bar-Zur et al. [19] introduced the WeRLman framework to study selfish mining under the volatile block reward model. They demonstrated that the security threshold (see Definition 8.2.5) of Bitcoin mining is lower in the volatile reward model compared to the fixed block reward model. They attributed this reduced security threshold in the transaction-fee setting to the arrival of

occasional transactions with exceptionally high transaction fees, known as *whale* transactions.

In Section 8.3, we argue that the WeRLman framework suffers from a *predictive capability* issue, where a miner can predict the arrival time of the next whale transaction. This leads to an underestimation of the security threshold in the WeRLman framework.

3.3.1.1 The Predictive Capability in the WeRLman Model

The WeRLman environment considers two types of transactions: normal transactions, each with a normalized fee of 1, and whale transactions, each with a normalized fee of $1 + F$. In the WeRLman model, at each decision-making point (i.e., when a new-height block is mined), the adversary knows in advance whether the next transaction added to the mempool will be a whale or a normal transaction, allowing it to decide on their next action accordingly. We refer to this ability to predict the type of the next transaction as *predictive capability*. This capability benefits the adversary by enabling more informed decisions on whether to publish or withhold their blocks. Note in reality, miners do not have advance knowledge about the nature of future transactions. While mining a new block, miners constantly update the contents of their blocks to include the highest-fee transactions available. As a result, any whale transactions present in the mempool are included in the already-mined block, implying that the mempool is cleared of whale transactions once a new block is mined.

3.3.1.2 Effect of Predictive Capability on Bitcoin Security Threshold

Our analysis in Section 8.3 reveals that the implicit predictive capability of the WeRLman model enables the adversary to make more strategic and profitable decisions about whether to publish or withhold its blocks. To illustrate the effect of predictive capability on the security threshold, we introduce a mining strategy $\pi_1^{\texttt{WeRLman}}$ specifically designed for the WeRLman environment (see Section 8.3.1). In this strategy, whenever the adversary mines a normal block, it withholds this block only if the subsequent transaction to be published is identified as a whale transaction. We demonstrate that following strategy $\pi_1^{\texttt{WeRLman}}$ in the WeRLman environment increases the adversary's profit compared to a real-world adversary, thereby reducing the security threshold.

We continue by explaining why, in a real-world Bitcoin environment, $\pi_1^{\texttt{WeRLman}}$ is not practical. A real-world adversary cannot limit block withholding only to scenarios where the next transaction is a whale, as it lacks any insight into the type of the next transaction. This suggests that the practical security threshold should be higher than what is measured under the WeRLman model.

We modify the WeRLman environment in Section 8.A2 to make it unpredictable. In Figure 8.2, we compare the security threshold for a profitable deviation from honest mining in the WeRLman environment and the non-predictable WeRLman environment. As shown in Figure 8.2, by making the WeRLman environment unpredictable, the reduction in the security threshold is less significant than it would be in the original WeRLman environment.

3.3.2 Simplified Volatile Reward Model

According to Figure 8.2, the unpredictable WeRLman model, although capable of anticipating a reduction in the security threshold in the long term, suggests that the security threshold remains unchanged until 2040. This imprecise result arises from the simplifications applied in the WeRLman model, which considers only two levels of fees for transactions: normal and whale.

In Section 8.4, we introduce a simplified volatile reward model equipped with multiple levels of fees, enabling a more accurate simulation of the real-world mempool. We developed an MDP-based tool to analyze mining strategies under the simplified model.

3.3.2.1 Simplified Bitcoin Mempool

We emphasize that the primary source of block reward volatility during the transaction fee era is not the occasional arrival of whale transactions with exceptionally high fees. Instead, analysis of mempool patterns reveals that block rewards are strongly correlated with *block generation times*. Longer block generation times allow more high-fee transactions to accumulate in the mempool, thereby increasing the block reward. In our mempool implementation, we shift the focus from whale transactions to block generation time as the key factor for analyzing Bitcoin mining strategies.

In Section 8.4.2, we present our simplified Bitcoin mempool implementation. To simulate the behavior of the mempool during a specific period, we first extract historical data [11] on Bitcoin block generation times and their corresponding total transaction fees for the target period. Using linear regression techniques, we then derive a *time-fee* equation for this period. This equation enables us to estimate the reward function and evaluate the profitability of various mining strategies. To constrain the state space, we utilize a discrete linear time-fee equation in our MDP environment. In this approach, the block generation time is divided into discrete steps, with each step corresponding to a reward determined by the time-fee equation.

3.3.2.2 MDP-Based Analysis under Simplified Volatile Reward Model

In Section 8.4.3, we develop an MDP tool to analyze mining strategies under the simplified Bitcoin mempool introduced in Section 3.3.2.1. The key difference between our MDP and other existing MDP tools in the literature is that our MDP tool stores the block generation times in the adversarial fork, allowing us to obtain the fee reward using the time-fee equation. Compared to the WeRLman model, our model incorporates a wider variety of blocks with different rewards, enabling a more fine-grained analysis of mining strategies.

Figure 8.4 compares the upper bound of the security thresholds obtained under our simplified model with that of the unpredictable WeRLman model. As shown in Figure 8.4, the increase in the number of discrete time points reduces the security threshold.

3.3.2.3 Profitability of Pred-DAM Selfish Mining in Transaction-Fee Era

In a fixed block reward model, no mining strategy can be more profitable than honest mining before difficulty adjustment, making deviation unprofitable before DAM, regardless of the adversarial mining share. However, as discussed in Section 8.5, by moving toward the transaction-fee era, deviation becomes immediately profitable before difficulty adjustment occurs. Figure 8.1 illustrates the Bitcoin security threshold before difficulty adjustment in the coming years under various communication capabilities (see Section 8.5 for details). As shown in Figure 8.1, when the protocol reward approaches 0, the Bitcoin security threshold drops to 16% for a typical communication capability of 50%, which is within the range of current mining pool shares.

3.3.3 Enhanced Volatile Reward Model

Despite improvements over existing models [19, 28] in the literature for analyzing Bitcoin's security in the transaction-fee era, the simplified volatile reward model introduced in Section 3.3.2 still has inherent limitations that hinder precise analysis of Bitcoin mempool patterns. Specifically, the simplified model assumes a linear relationship between time and accumulated fees, which is often inaccurate, as real-world data typically shows a concave pattern. It also relies solely on the generation time of the recent block as the dependent variable, overlooking the influence of preceding blocks' fees and generation times. Additionally, the use of sparse block data reduces prediction accuracy and leads to deviations from actual real-time behavior.

In Section 8.6, we introduce our Asynchronous Advantage Actor Critic (A3C)-based implementation designed to derive a near-optimal mining policy within an

enhanced volatile reward model, which addresses the limitations of the simplified model.

3.3.3.1 Enhanced Mempool Implementation

In Section 8.6.1, we introduce an enhanced mempool implementation that closely simulates the behavior of Bitcoin's actual mempool. A Bitcoin transaction fee is typically expressed in satoshis per virtual byte (sat/vByte), where each satoshi equals 10^{-8} BTC. Each Bitcoin block has a size limit of 1 virtual megabyte (vMB), and miners prioritize transactions offering the highest fees in satoshis per virtual byte to maximize value. Our implementation categorizes the transactions in the mempool into N^{range} sat/vByte ranges. For each fee range, we predict a *weight-time* equation that estimates transaction weight growth within that range. To replicate Bitcoin mempool patterns for a given period, we extract fine-grained Bitcoin mempool data and apply curve fitting (polynomial and logarithmic regressions) to derive weight-time equations for all fee ranges. As time progresses in the A3C environment, new transactions are added to the mempool based on the weight-time equations. Whenever a block is mined, the miner can choose to fill the block with available transactions, mimicking the behavior of the real-world Bitcoin mempool.

3.3.3.2 States and Actions in Our A3C-Based Implementation

Section 8.6.2 presents the state representation and action set used in our implementation. Each state in our implementation, in addition to information about the fork lengths, includes the mempool views corresponding to all the blocks involved in the fork race. Note that the mempool view from the perspective of each block differs, as each block modifies the state of the mempool. Storing the mempool view with respect to all the blocks is necessary because the fork race is ongoing, and it is not yet known which blocks will be added to the canonical chain. The action set in our implementation includes a wide range of strategies, such as `wait`, `match`, `override`, `adopt`, `block undercut`, and `fork undercut`.

3.3.3.3 Evaluation of the A3C-Based Implementation

In Section 8.6.4, we present the profitability outcomes of various mining strategies derived from our A3C-based implementation. These results are evaluated across a wide range of days to capture Bitcoin's security under diverse mempool conditions. Our findings show that the near-optimal strategy learned by the A3C model can outperform the profitability of the strategy obtained

through the MDP framework. Our analysis shows that as block rewards continue to decrease, miners have a stronger incentive to deviate from the honest strategy, which could pose a risk to Bitcoin's long-term security. Maintaining a higher base transaction fee in the mempool can help mitigate mining attacks, particularly as the network transitions to the transaction-fee era. Our results also indicate that on days with lower base fees, the risk of undercutting attacks (see Section 8.A1.2) increases.

Chapter 4

Incentive Analysis in PoS Blockchains

4.1 Selfish Proposing in Longest-Chain Proof-of-Stake Protocols

This section provides an overview of our work in [87], which is presented in Chapter 9. The underlying fork choice rule of Bitcoin, namely the longest-chain rule, is also implemented in several Proof-of-Stake (PoS) protocols. Examples include the Ouroboros protocol in Cardano [33, 62], the Emmy protocol in Tezos [52], and Snow White [30]. The incentivization mechanisms of these longest-chain PoS (LC-PoS) protocols, similar to that of Bitcoin, are vulnerable to the selfish mining attack. However, since there is no mining in PoS, we refer to this attack in the context of PoS blockchains as *selfish proposing*.

In [87], we analyze the impact of selfish proposing in the context of longest-chain PoS protocols. We demonstrate that inherent PoS-specific factors, such as the nothing-at-stake phenomenon and predictability, make the selfish proposing attack in PoS blockchains more destructive than its counterpart in Bitcoin. We categorize longest-chain Proof-of-Stake (LC-PoS) protocols based on their level of proposer predictability into three groups: 1) *Perfect-randomness protocols*, in which the block proposer for a slot is not known to anyone before the slot begins; 2) *Semi-predictable protocols*, typically based on a verifiable random function (VRF), where the block proposer is known only to the proposer themselves before the slot starts; and 3) *Fully-predictable protocols*, in which the block proposer for each slot is known in advance to all validators. Our analysis shows

that as the level of proposer predictability increases in a PoS protocol, selfish proposing becomes more profitable.

Our main contributions in [87] are as follows:

- Proposing a selfish proposing strategy for LC-PoS protocols with perfect randomness that outperforms the standard selfish mining strategy (Section 4.1.1).
- Providing a theoretical analysis of selfish proposing in fully predictable LC-PoS protocols (Section 4.1.2).
- Developing a Deep Q-Learning (DQL) tool to analyze selfish proposing in practical semi-predictable LC-PoS protocols (Section 4.1.3).

4.1.1 Selfish Proposing in LC-PoS Protocols with Perfect Randomness

If, in an LC-PoS protocol, the proposer of a slot is not known to anyone before the slot begins, the protocol is said to achieve the perfect-randomness property [44]. This property ensures the same level of non-predictability as that of PoW protocols such as Bitcoin, where the next block miner is not known until the block is mined. However, none of the PoS protocols implemented in practice fully achieve this property, as it requires the blockchain to rely on an external random seed revealed at the start of each slot.

Although an adversary cannot benefit from proposer predictability in LC-PoS protocols with perfect randomness, it can still exploit the nothing-at-stake phenomenon to launch a more profitable attack than is possible in PoW settings. In Section 9.2, we present a generalized framework for analyzing nothing-at-stake selfish proposing in LC-PoS protocols with perfect randomness.

4.1.1.1 Nothing-at-Stake Phenomenon

In PoS protocols, the nothing-at-stake problem refers to the phenomenon in which a slot's selected block proposer can build multiple blocks on top of different forks at no cost [22]. This issue does not exist in PoW blockchains, where mining on any fork requires significant computational resources. If the nothing-at-stake problem is left unaddressed, it can undermine the security of PoS protocols. Most permissionless PoS blockchains, including Ethereum [9], Solana [98], and Polkadot [97], use slashing mechanisms to mitigate this risk. Since producing multiple blocks for the same slot is a detectable form of misbehavior, the stake of any validator caught doing so is subject to being slashed or burned.

4.1.1.2 Impact of Nothing-at-Stake on Selfish Proposing

Even if nothing-at-stake misbehavior is slashable in an LC-PoS protocol, an adversary can still exploit it to intensify selfish proposing. Due to the nothing-at-stake phenomenon, a selfish proposer in an LC-PoS protocol can easily modify the structure of its secret fork by changing the canonical block on which the fork is built. This structure modification cannot be slashed since the adversary does not publish the earlier version of its secret fork, making the behavior undetectable.

In Section 9.2, we extend the set of possible actions proposed for optimal selfish mining in PoW [85], as well as the selfish mining strategy introduced in [44] for PoS, by introducing a generalized set of new actions enabled by the nothing-at-stake phenomenon.

4.1.1.3 Security Threshold in LC-PoS with Perfect Randomness

In Section 9.2.3, we theoretically analyze the minimum stake share required for profitable selfish proposing——referred to as the security threshold——in LC-PoS protocols with perfect randomness. While the security threshold for profitable selfish mining in PoW protocols has been proven to be 25% for an adversary with a normal communication capability [43, 85], we show that this threshold is lower than 24.198% in LC-PoS protocols with perfect randomness.

4.1.2 Selfish Proposing in Full-Predictable LC-PoS

Selfish mining in the context of Bitcoin is considered a risky attack because if the adversary initiates a fork race, there is a possibility of losing the race and orphaning its own blocks. However, in a full-predictable LC-PoS system, a selfish proposer can exploit the complete predictability of future block proposers to launch a fork race that is guaranteed to succeed. This predictability allows the adversary to perform selfish proposing without losing adversarial blocks while maximizing the number of orphaned non-adversarial blocks. In Section 9.3.1, we present a novel theoretical approach based on random walks on a grid to analyze full-predictable selfish proposing.

4.1.2.1 Random Walks on a Grid

In Sections 9.A4.2, 9.A5, and 9.A6, we present a theoretical method based on random walks to analyze the block ratio achieved under several block-proposing strategies. In this approach, a two-dimensional (x, y)-grid is used to represent the race between the honest and adversarial forks, as illustrated in Figure 4.1.

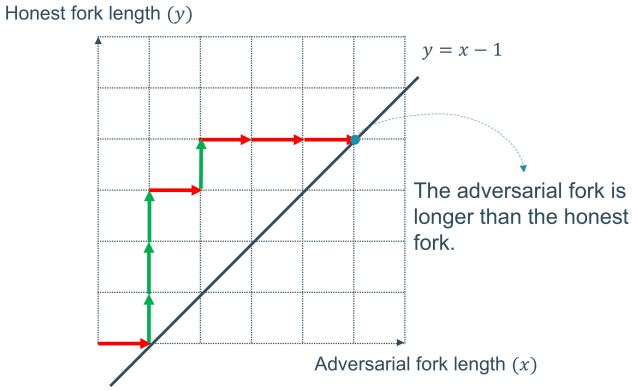


Figure 4.1: A block path on a 2-D grid. Red and green arrows represent adversarial and honest slots, respectively. If the block path reaches the line $y = x - 1$, the adversarial fork becomes longer than the honest one, indicating that the adversary wins the fork race.

Any given sequence of block proposers can be mapped to a path on this grid, referred to as the block path. When the block proposer for a time slot is honest, the block path moves one step up; when the proposer is adversarial, it moves one step to the right. This framework allows us to compute the probability of winning a fork race, the expected number of canonical blocks, and the expected number of orphaned blocks under various strategies.

4.1.2.2 Optimal Full-Predictable Selfish Proposing

In Section 9.A7, we propose the optimal selfish proposing strategy in a full-predictable LC-PoS blockchain for an adversary with communication capability (see Definition 9.A3.2) equal to 0. This strategy identifies honest checkpoint slots (see Definition 9.3.1), whose proposed blocks cannot be orphaned by any adversarial fork initiated before the slot. Then, for each pair of consecutive checkpoints, it searches for the optimal formation of adversarial forks that maximizes the number of orphaned honest blocks.

4.1.3 Deep Q-Learning for Analyzing Selfish Proposing

Theoretical analysis is useful for examining selfish proposing in straightforward settings, such as fully predictable LC-PoS protocols. However, as the environment becomes more complex, relying solely on theoretical methods becomes increasingly challenging. This is especially true in semi-predictable

LC-PoS protocols (such as Cardano) that use a VRF to select block proposers, where each slot can have none, one, or multiple potential proposers. To address this complexity, Section 9.4 introduces a tool designed using deep Q-learning (DQL) to effectively analyze selfish proposing behavior in these more complex semi-predictable LC-PoS environments.

4.1.3.1 DQL Implementation

The environment implemented in our DQL tool is capable of simulating the behavior of various types of LC-PoS protocols, including semi-predictable, fully predictable, and those with perfect randomness. In predictable protocols, the state representation includes not only information about the current forks but also the future block proposers. In a semi-predictable protocol, this future proposer information is limited to slots where the proposer is the adversary itself. The action space for the agent (adversary) in our implementation corresponds to the generalized action set introduced in Section 9.2, which extends the standard PoW actions to include those enabled by the nothing-at-stake phenomenon.

4.1.3.2 Quantifying the Selfish Proposing Attack

Our analysis confirms that the security threshold for a profitable selfish proposing attack in both semi-predictable and fully predictable protocols reduces to zero [22]. This is because knowledge of future slot proposers enables the adversary to initiate forks only when it is certain to win the fork race.

Figures 9.4a and 9.4b compare the block ratios achieved through selfish proposing/mining using our DQL implementation across various protocols: PoW, LC-PoS with perfect randomness, semi-predictable LC-PoS, and fully predictable LC-PoS. Our analysis shows that, while the nothing-at-stake phenomenon has only a minor impact on a selfish proposer's block ratio, predictability can significantly increase it, particularly when the attacker's communication capability is relatively low. Specifically, with an adversarial stake or mining share of 1/3 and zero communication capability, the block ratios observed under selfish proposing/mining are as follows: 0.337 for PoW, 0.342 for LC-PoS with perfect randomness, 0.3938 for semi-predictable LC-PoS, and 0.4383 for fully predictable LC-PoS. We conclude that the incentivization mechanism in LC-PoS protocols is more susceptible to fairness loss compared to that of PoW blockchains.

4.2 Commitment Attacks on Ethereum's Reward Mechanism

This section provides an overview of our work in [89], which is presented in Chapter 10. Ethereum's underlying fork choice rule, known as LMD-GHOST, selects the chain with the most validator support, measured by the highest number of votes, as the canonical chain. To incentivize participation, Ethereum rewards validators for submitting votes to the network.

In [89], we identify a vulnerability in Ethereum's reward mechanism that, if exploited, allows an adversarial validator to launch a reorganization attack on the Ethereum protocol and gain an unfair share of rewards. The vulnerability arises from the centralized design of the reward mechanism, where the block proposer of a given slot has full control over the head vote rewards for the attestors of the previous slot. Our contributions in [89] are as follows:

- Proposing several reorganization attacks, ranging from single-slot to long-range, that exploit a vulnerability in Ethereum's head vote reward mechanism (Section 4.2.1).
- Introducing a Directed Acyclic Graph (DAG) vote-based reward mechanism to achieve a more decentralized and fairer distribution of rewards (Section 4.2.2).

4.2.1 Commitment Attacks: Overview and Causes

Ethereum validators are typically rational players who aim to maximize their incentives. The rationality assumption of validators implies that they may deviate from the expected honest behavior if an alternative strategy offers higher rewards. In this section, we introduce a set of attacks based on the assumption of a rational majority, where an adversarial block proposer can manipulate the incentives of other validators to act according to the adversary's desired strategy.

4.2.1.1 Head Vote Reward Mechanism in Ethereum

The vote submitted for LMD-GHOST to determine the canonical chain is called a head vote. To be eligible for a reward, a head vote must be both correct and timely. A head vote for slot t is considered correct if it supports the latest block included in the canonical chain up to and including slot t. It is timely if it is included in the block of slot $t+1$ [41]. A detailed explanation of Ethereum's reward mechanism is provided in Section 10.A1.1.

The correctness of a head vote is crucial, as incorrect or arbitrary votes can prevent the majority from reaching consensus. Timeliness is also critical, as it enables the proposer of the next slot to correctly identify the tip of the chain when proposing their block.

4.2.1.2 Centralization in the Allocation of Head Vote Rewards

According to Ethereum's current reward mechanism, the timeliness of a head vote for slot t is determined by a single entity: the block proposer of slot $t+1$. This implies that if the slot $t+1$ proposer excludes some of the votes submitted by attestors of the previous slot from its block, those votes can no longer be proven timely and, as a result, become ineligible for rewards.

4.2.1.3 The Simple Reorg Attack in the Presence of Solo Validators

An adversarial validator can exploit the centralization in Ethereum's reward mechanism to conduct a reorganization attack. In Section 10.4.1, we introduce the simplest form of this attack, which, if successfully executed, results in a single-slot reorg.

In this attack, as depicted in Figure 10.1b, the adversarial block proposer of slot $t+1$ requests the attestors of slot t to vote for the parent of the slot t block instead of the block itself. The adversarial proposer threatens to exclude any non-compliant attestors from its block. If a majority of attestors follow the adversary's strategy, the attack succeeds, meaning the block at slot t fails to gather sufficient support and is subsequently reorged. In Section 10.4.1, we prove that a Nash equilibrium exists in which the simple attack succeeds in the presence of solo validators.

4.2.1.4 The Strong Simple Reorg Attack in the Presence of Solo Validators

In the simple game introduced earlier, compliance with the adversary is a weakly dominant strategy for attestors. If the majority of slot t attestors do not follow the adversary's strategy, an individual attestor becomes indifferent between compliance and non-compliance, as they receive no reward in either case.

To address this, we propose a strong simple attack in Section 10.5, in which compliance with the adversary becomes a strongly dominant strategy. This means that an attestor is always incentivized to follow the attack strategy, regardless of the actions of other attestors.

4.2.1.5 The Extended Reorg Attack in the Presence of Solo Validators

In Section 10.6, we introduce an extended attack designed to reorg a sequence of consecutive Ethereum blocks using *transaction-less* blocks. This attack threatens the liveness property of the Ethereum chain and enables the adversary to eventually produce highly valuable blocks following a prolonged period of empty blocks.

The core idea is that the adversarial block proposer at slot $t+1$ coerces attestors from several preceding slots to vote for the latest transaction-less block available in their view. This is achieved by threatening to exclude their votes from the next slot's block if they fail to comply. Similarly, the adversary pressures earlier block proposers to produce transaction-less blocks by threatening that non-compliant blocks will not receive sufficient attestations for inclusion in the canonical chain.

4.2.1.6 Reorg Attacks in the Presence of Staking Pools

In Section 10.7, we analyze both the simple and extended attacks in the context of staking pools. We define a staking pool as a group of rational, colluding solo validators distributed across different slots. We prove the existence of a Nash equilibrium for both the simple and extended attacks in which the adversary succeeds. These proofs rely on the assumption that the number of solo validators per slot controlled by a staking pool remains below Ethereum's proposer boost threshold, currently set at 40% of the slot's attestor set.

4.2.1.7 Selfish Mining-Inspired Attack

The analysis of the extended game in the presence of staking pools shows that when staking pools collude to coordinate strategies for a set of validators exceeding the proposer boost threshold, compliance with the adversary's rules is no longer a dominant strategy. As a result, staking pool collusion mitigates the extended attack. In Section 10.7.3, we introduce a selfish mining-inspired attack in which even colluding validators remain incentivized to follow the adversary's strategy.

This attack is effective during time windows where the number of adversarial slots is greater than or equal to the number of non-adversarial slots. In such cases, the adversarial staking pool can leverage its private fork to reorg the non-adversarial blocks within the window as in the original selfish mining attack.

4.2.2 New Vote Reward Mechanism

In Section 10.8, we propose a new reward mechanism inspired by a Directed Acyclic Graph (DAG) topology to promote more decentralized reward distribution. This mechanism not only mitigates the attacks described in Section 4.2.1, but also improves fairness. Under the current design, when a slot is missed, i.e., no block is proposed, attestors from the previous slot lose their head vote rewards due to the failure of the next slot's proposer. Our proposed mechanism addresses this issue by ensuring head vote rewards are preserved, even when a minority of attestors are offline.

4.2.2.1 DAG Votes Mechanism

In the proposed DAG-based voting mechanism introduced in Section 10.8, all slot $t+1$ attestors are required to sign the slot t votes visible in their view at the start of slot $t+1$. A signature from a slot $t+1$ attestor on a slot t vote is referred to as an evidence. These evidences must be included in the canonical chain. Under this new reward mechanism, a vote is considered timely if it collects a sufficient number of evidences from attestors in the following slot.

This approach shifts the power to determine vote timeliness from the next slot's proposer to a committee of validators. As a result, the adversary's control over vote rewards is reduced, unless it controls a significant portion of the committee.

In Section 10.8.1, we present a security analysis of the LMD-GHOST protocol combined with our new reward mechanism. We prove that, under a rational setting, there exists a subgame perfect Nash equilibrium in which the protocol achieves security (namely, safety and liveness).

4.2.2.2 A Practical Version of DAG Votes Mechanism

Given the large committee size of attestors per slot in Ethereum, the solution proposed in Section 4.2.2.1 introduces significant communication and storage overhead, making it impractical for direct implementation. To address this challenge, in Section 10.8.2, we propose a practical variant of the DAG votes mechanism. In this version, a subset of attestors is randomly selected to act as slot aggregators, who are responsible for generating evidences for the previous slot's votes. Under synchrony, these evidences can be aggregated into a single signature, significantly reducing overhead.

In Sections 10.A6 and 10.A7, we analyze the block space, computational, and communication overheads of the new reward scheme under both the optimistic case of synchrony and the worst case of asynchrony. These analyses also provide bounds on the number of aggregators per slot required to maintain security.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Permissionless blockchains are one of the most fascinating technological innovations of the 21st century. They facilitate decentralized consensus, paving the way for a wide range of applications, such as digital cash, decentralized finance (DeFi), immutable record-keeping, auction mechanisms, and voting systems. Like any technology, permissionless blockchains rely on critical security pillars that are essential to their advancement. Compromising these pillars could result in the collapse of both the blockchains and the applications built upon them.

In this thesis, we analyzed the consensus layer security of several prominent permissionless blockchains, including Bitcoin, Ethereum, and Cardano. The primary focus was to highlight the critical role of incentivization (reward) mechanisms in the design of secure blockchains. While often overlooked in security analyses of permissionless blockchains, participants in these systems are typically rational actors seeking to maximize their incentives. This indicates that applying the same security analysis used for permissioned blockchains——where the majority is assumed to act honestly and follow the prescribed strategy——may be inadequate for studying the security of permissionless blockchains. Such an approach fails to consider the behavior of payoff-maximizing participants who adopt strategies based on their own interests rather than the intentions of the protocol designer. An unfair or centralized reward mechanism can undermine the incentive compatibility of a permissionless blockchain, potentially leading to major security vulnerabilities or chaotic scenarios where participants deviate arbitrarily from the prescribed strategy.

We began by discussing the role of Bitcoin's difficulty adjustment mechanism in the profitability of mining strategies. We demonstrated that incorporating orphan blocks into difficulty adjustment can effectively make mining power destruction attacks unprofitable for adversaries. However, this mitigation also reduces profitability for honest miners and becomes ineffective against attacks that do not generate orphan blocks. We then expanded the Bitcoin security analysis to include semi-rational mining pools. We showed that, in a rational setting, a malicious pool with mining shares lower than the well-known security thresholds can still target smaller pools, suggesting that an unfair distribution of mining shares and large disparities between mining pools pose a significant threat to Bitcoin's security. We also examined the impact of Bitcoin's reward halving on its future security. We demonstrated that the volatility of Bitcoin block rewards, caused by fluctuating block generation times, can lower the security threshold for immediately profitable deviations. In the near future, if transaction fees cannot provide a consistent and stable income for miners, Bitcoin may face serious security risks, with miners potentially acting maliciously to compensate for their invested resources.

In this thesis, we also analyzed the security of longest-chain PoS protocols, such as Cardano, against selfish mining attacks. We demonstrated that the proposer predictability inherent in practical PoS protocols allows adversaries to strategically launch attacks during moments of greater certainty about their success, thereby minimizing the risk of their blocks being orphaned. Our analysis revealed that longest-chain PoS protocols are more susceptible to malicious deviations from the honest strategy compared to PoW protocols. Additionally, we examined the consensus mechanism of the Ethereum blockchain, which relies on the LMD-GHOST protocol for fork choice. We demonstrated that a malicious Ethereum block proposer can exploit the centralization of Ethereum's reward mechanism to incentivize rational voters to support the adversarial strategy, enabling reorganization attacks. Recognizing the need for a more decentralized reward mechanism in Ethereum, we proposed a DAG-based voting solution that establishes a fairer reward system capable of mitigating these incentive-based attacks.

5.2 Future Work

The ever-evolving field of permissionless blockchains continually introduces new protocols and innovations, each accompanied by potential security concerns. This necessitates ongoing efforts to identify vulnerabilities and attack vectors, as well as to update and refine protocols to ensure continuous progress toward more secure and decentralized blockchains. Below, we outline potential studies aimed

at enhancing the security and incentive compatibility of prominent existing permissionless blockchains.

Modify Bitcoin's longest chain rule to enhance security against selfish mining. The primary reason Bitcoin's longest chain is vulnerable to selfish mining lies in the probabilistic nature of mining, where a pool with a minority of mining power can occasionally experience short periods of luck, controlling most of the blocks. Our objective is to enhance the Bitcoin protocol to significantly reduce the likelihood of a block proposer mining a longer chain, even within a short time frame. The core concept is to require miners to publish and include intermediary mini-PoW blocks between actual Bitcoin blocks. These mini-PoW blocks are much easier to mine than Bitcoin blocks but still contribute to the weight of the chain they are attached to. Incorporating this fine-grained PoW information into the chain more accurately reflects the distribution of mining power, thereby reducing the chances of a malicious miner succeeding in selfish mining.

Similar methods have been proposed in previous work, such as [84] and [51], to make the Bitcoin protocol more resistant to block reorganization. However, these proposals have proven ineffective against selfish mining [99]. We believe this ineffectiveness arises from design flaws in defining the difficulty, weight, and reward of a mini-PoW block. Our goal is to refine these design elements to develop an effective defense against selfish mining.

Design of incentive-based secure superlight clients for Bitcoin. The Bitcoin protocol supports Simplified Payment Verification (SPV) clients, which download only the headers of Bitcoin blocks to verify the longest chain in the network [74]. Although block headers are significantly smaller than full blocks, the communication overhead for SPV clients grows linearly with the length of the blockchain. To reduce the download size, several approaches like Superblock [60] and FlyClient [24] have been proposed in the literature to support superlight clients, enabling them to operate with only polylogarithmic communication relative to the total number of blocks. However, these proposals typically require modifications to the underlying consensus protocol, such as the adoption of a soft fork, to be practical, which limits their deployability [61]. One potential solution to this challenge is the design of superlight client protocols whose security relies on economic incentives. These incentive-based approaches allow a single Bitcoin full node (regardless of network-wide compliance) to generate superlight proofs of PoW, with their security rooted in on-chain Bitcoin deposits.

Design of an efficient and decentralized vote-rewarding scheme for Ethereum. The Ethereum protocol is transitioning toward single-slot finality (shorter finality period), where all validators are expected to vote in each slot [14, 31]. Including all of these votes in the block can significantly increase

block size [1]. Another consequence of moving to single-slot finality is that the introduced commitment attacks become more destructive, as they can compromise a larger portion of the attestors' rewards. This is due to the timeliness window for finalization votes shrinking from one epoch to one slot (similar to the timeliness window of LMD-GHOST votes in current Ethereum), which allows a malicious proposer to exploit both LMD-GHOST and finalization votes.

These concerns highlight the need to address the centralization and inefficiency in the current reward mechanism. While our proposed DAG vote solution in this thesis is decentralized, it suffers from significant communication and block space overhead, especially during asynchrony and in the era of single-slot finality. We aim to improve this solution by incorporating more efficient design choices, such as using zero-knowledge proofs to generate succinct evidence of participation in the voting process.

Incentive compatibility analysis of emerging responsibilities in Ethereum. The Ethereum blockchain is constantly evolving with new updates that define new roles and responsibilities, each of which should be properly incentivized. One of the most recent innovations is the use of blobs to improve scalability [8, 64]. A blob is a large chunk of data linked to an Ethereum block, used to store off-chain information, such as rollup transaction data. Although the Ethereum virtual machine does not process blob data, Ethereum validators must reach consensus on the publication of the entire blob, ensuring no part is missing.

Each Ethereum validator is responsible for downloading small parts of these blobs and temporarily storing them to verify their availability. However, the Ethereum protocol does not provide specific incentives for storing the blob data. Aside from complex entities that can provide blob data services for Ethereum and rollup users, a typical Ethereum validator has little incentive to store this data, even for a short period. This creates a potential risk of centralizing the data availability layer, making it vulnerable to DoS attacks. Our goal is to improve the incentive compatibility of the data availability layer to enhance its economic security.

[1] With nearly 1 million Ethereum validators, the participation bitfield encoding of the aggregated signature alone would be around 125 KB, exceeding the current size of an Ethereum block with all transactions.

Bibliography

[1] Altair. https://github.com/ethereum/consensus-specs/blob/dev/specs/altair/beacon-chain.md. Accessed: 2025.

[2] Average Bitcoin block size. https://www.blockchain.com/explorer/charts/avg-block-size. Accessed: 2025.

[3] Bitcoin block difficulty. https://en.bitcoin.it/wiki/Difficulty. Accessed: 2025.

[4] Bitcoin block header. https://developer.bitcoin.org/reference/block_chain.html. Accessed: 2025.

[5] Bitcoin controlled supply. https://en.bitcoin.it/wiki/Controlled_supply. Accessed: 2025.

[6] Cardano developer portal. https://developers.cardano.org/. Accessed: 2025.

[7] Cardano documentation. https://docs.cardano.org/. Accessed: 2025.

[8] Eip-4844: Shard blob transactions (proto-danksharding). https://github.com/ethereum/EIPs/blob/master/EIPS/eip-4844.md. Accessed: 2025.

[9] Ethereum developer resources. https://ethereum.org/en/developers/. Accessed: 2025.

[10] Ethereum proof-of-stake (the Merge). https://ethereum.org/en/upgrades/merge/. Accessed: 2025.

[11] Historical data on block transaction fees and generation times. https://blockchair.com/. Accessed: 2025.

[12] Hyperledger besu. https://besu.hyperledger.org/. Accessed: 2025.

[13] Mempool, pool, and blockchain explorer for Bitcoin. https://mempool.space/. Accessed: 2025.

[14] Single slot finality. https://ethereum.org/en/roadmap/single-slot-finality/. Accessed: 2025.

[15] Understanding Cardano monetary policy. https://cexplorer.io/article/understanding-cardano-monetary-policy. Accessed: 2025.

[16] Giuseppe Ateniese, Ilario Bonacina, Antonio Faonio, and Nicola Galesi. Proofs of space: When space is of the essence. In *Security and Cryptography for Networks*, pages 538–557. Springer, 2014.

[17] Christian Badertscher, Peter Gaži, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 913–930, 2018.

[18] Maryam Bahrani, Pranav Garimidi, and Tim Roughgarden. Centralization in block-building and proposer-builder separation. In *International Conference on Financial Cryptography and Data Security*, pages 331–349. Springer, 2024.

[19] Roi Bar-Zur, Ameer Abu-Hanna, Ittay Eyal, and Aviv Tamar. WeRLman: To tackle whale (transactions), go deep (RL). In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 93–110. IEEE, 2023.

[20] Roi Bar-Zur, Danielle Dori, Sharon Vardi, Ittay Eyal, and Aviv Tamar. Deep bribe: Predicting the rise of bribery in blockchain mining with deep RL. In *2023 IEEE Security and Privacy Workshops*, pages 29–37. IEEE, 2023.

[21] Joseph Bonneau. Why buy when you can rent? Bribery attacks on Bitcoin-style consensus. In *International Conference on Financial Cryptography and Data Security*, pages 19–26. Springer, 2016.

[22] Jonah Brown-Cohen, Arvind Narayanan, Alexandros Psomas, and S. Matthew Weinberg. Formal barriers to longest-chain proof-of-stake protocols. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 459–473, 2019.

[23] Eric Budish, Andrew Lewis-Pye, and Tim Roughgarden. The economic limits of permissionless consensus. In *Proceedings of the 25th ACM Conference on Economics and Computation*, pages 704–731, 2024.

[24] Benedikt Bünz, Lucianna Kiffer, Loi Luu, and Mahdi Zamani. Flyclient: Super-light clients for cryptocurrencies. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 928–946. IEEE, 2020.

[25] Vitalik Buterin. Proposal for mitigation against balancing attacks to LMD GHOST. https://notes.ethereum.org/@vbuterin/lmd_ghost_mitigation, 2020.

[26] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*, 2017.

[27] Vitalik Buterin, Diego Hernandez, Thor Kamphefner, Khiem Pham, Zhi Qiao, Danny Ryan, Juhyeok Sin, Ying Wang, and Yan X Zhang. Combining GHOST and Casper. *arXiv preprint arXiv:2003.03052*, 2020.

[28] Miles Carlsten, Harry Kalodner, S. Matthew Weinberg, and Arvind Narayanan. On the instability of Bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 154–167, 2016.

[29] Miguel Castro and Barbara Liskov. Practical Byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186. USENIX, 1999.

[30] Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In *Financial Cryptography and Data Security*, pages 23–41. Springer, 2019.

[31] Francesco D'Amato, Roberto Saltini, Thanh-Hai Tran, and Luca Zanolini. 3-slot-finality protocol for Ethereum. *arXiv preprint arXiv:2411.00558*, 2024.

[32] Francesco D'Amato and Luca Zanolini. Recent latest message driven GHOST: Balancing dynamic availability with asynchrony resilience. In *2024 IEEE 37th Computer Security Foundations Symposium (CSF)*, pages 127–142. IEEE, 2024.

[33] Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Advances in Cryptology–EUROCRYPT 2018*, volume 10821, pages 66–98. Springer, 2018.

[34] Primavera De Filippi and Aaron Wright. *Blockchain and the law: The rule of code*. Harvard University Press, 2018.

[35] Evangelos Deirmentzoglou, Georgios Papakyriakopoulos, and Constantinos Patsakis. A survey on long-range attacks for proof of stake protocols. *IEEE access*, 7:28712–28725, 2019.

[36] Amir Dembo, Sreeram Kannan, Ertem Nusret Tas, David Tse, Pramod Viswanath, Xuechao Wang, and Ofer Zeitouni. Everything is a race and Nakamoto always wins. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 859–878, 2020.

[37] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2):288–323, 1988.

[38] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In *Annual Cryptology Conference (CRYPTO)*, volume 9216, pages 585–605. Springer, 2015.

[39] Francesco D'Amato, Joachim Neu, Ertem Nusret Tas, and David Tse. Goldfish: No more attacks on Ethereum?! In *International Conference on Financial Cryptography and Data Security*, pages 3–23. Springer, 2024.

[40] Ben Edgington. LMD Ghost. https://eth2book.info/capella/part2/consensus/lmd_ghost/. Accessed: 2025.

[41] Ben Edgington. Reward of the Ethereum consensus layer. https://eth2book.info/capella/part2/incentives/rewards/. Accessed: 2025.

[42] Ben Edgington. *A technical handbook on Ethereum's move to proof of stake and beyond*. 2023. https://eth2book.info/.

[43] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International Conference on Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.

[44] Matheus V.X. Ferreira and S. Matthew Weinberg. Proof-of-stake mining games with perfect randomness. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 433–453, 2021.

[45] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[46] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin backbone protocol with chains of variable difficulty. In *Annual International Cryptology Conference*, volume 10401, pages 291–323. Springer, 2017.

[47] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin backbone protocol: Analysis and applications. *Journal of the ACM*, 71(4):1–49, 2024.

[48] Peter Gaži, Aggelos Kiayias, and Alexander Russell. Tight consistency bounds for Bitcoin. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 819–838, 2020.

[49] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling Byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68, 2017.

[50] Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *Acm Sigact News*, 33(2):51–59, 2002.

[51] Xueping Gong, Qing Zhang, Huizhong Li, and Jiheng Zhang. Improving blockchain consistency bound by assigning weights to random blocks. *Operations Research*, 73(4):2156–2176, 2025.

[52] L. M. Goodman. Tezos: A self-amending crypto-ledger. https://tezos.com/whitepaper.pdf, 2014.

[53] Cyril Grunspan and Ricardo Pérez-Marco. On profitability of selfish mining. *arXiv preprint arXiv:1805.08281*, 2018.

[54] Cyril Grunspan and Ricardo Pérez-Marco. Profit lag and alternate network mining. In *The International Conference on Mathematical Research for Blockchain Economy*, pages 115–132. Springer, 2023.

[55] Lioba Heimbach, Lucianna Kiffer, Christof Ferreira Torres, and Roger Wattenhofer. Ethereum's proposer-builder separation: Promises and realities. In *Proceedings of the 2023 ACM on Internet Measurement Conference*, pages 406–420, 2023.

[56] Yuming Huang, Jing Tang, Qianhao Cong, Andrew Lim, and Jianliang Xu. Do the rich get richer? Fairness analysis for blockchain incentives. In *Proceedings of the 2021 International Conference on Management of Data*, pages 790–803, 2021.

[57] Dragos I Ilie, Sam M Werner, Iain D Stewart, and William J Knottenbelt. Unstable throughput: When the difficulty algorithm breaks. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–5. IEEE, 2021.

[58] Aljosha Judmayer, Nicholas Stifter, Alexei Zamyatin, Itay Tsabary, Ittay Eyal, Peter Gazi, Sarah Meiklejohn, and Edgar Weippl. Pay to win: Cheap, crowdfundable, cross-chain algorithmic incentive manipulation attacks on PoW cryptocurrencies. Cryptology ePrint Archive, Paper 2019/775, 2019.

[59] Aljosha Judmayer, Nicholas Stifter, Alexei Zamyatin, Itay Tsabary, Ittay Eyal, Peter Gaži, Sarah Meiklejohn, and Edgar Weippl. Sok: Algorithmic incentive manipulation attacks on permissionless PoW cryptocurrencies. In *Financial Cryptography and Data Security: FC 2021 International Workshops*, volume 12675, pages 507–532. Springer, 2021.

[60] Aggelos Kiayias, Andrew Miller, and Dionysis Zindros. Non-interactive proofs of proof-of-work. In *Financial Cryptography and Data Security*, pages 505–522. Springer, 2020.

[61] Aggelos Kiayias, Andrianna Polydouri, and Dionysis Zindros. The velvet path to superlight blockchain clients. In *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, pages 205–218, 2021.

[62] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Advances in Cryptology – CRYPTO 2017*, volume 10401, pages 357–388. Springer, 2017.

[63] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. https://peercoin.net/assets/paper/peercoin-paper.pdf, 2012.

[64] Michał Król, Onur Ascigil, Sergi Rene, Etienne Rivière, Matthieu Pigaglio, Kaleem Peeroo, Vladimir Stankovic, Ramin Sadre, and Felix Lange. Data availability sampling in Ethereum: Analysis of P2P networking requirements. *arXiv preprint arXiv:2306.11456*, 2023.

[65] Yujin Kwon, Hyoungshick Kim, Jinwoo Shin, and Yongdae Kim. Bitcoin vs. Bitcoin Cash: Coexistence or downfall of Bitcoin Cash? In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 935–951. IEEE, 2019.

[66] Leslie Lamport. The part-time parliament. *ACM Transactions on Computer Systems (TOCS)*, 16(2):133–169, 1998.

[67] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[68] Andrew Lewis-Pye and Tim Roughgarden. Resource pools and the CAP theorem. *arXiv preprint arXiv:2006.10698*, 2020.

[69] Andrew Lewis-Pye and Tim Roughgarden. Permissionless consensus. *arXiv preprint arXiv:2304.14701*, 2023.

[70] Kevin Liao and Jonathan Katz. Incentivizing blockchain forks via whale transactions. In *Financial Cryptography and Data Security: FC 2017 International Workshops*, pages 264–279. Springer, 2017.

[71] Patrick McCorry, Alexander Hicks, and Sarah Meiklejohn. Smart contracts for bribing miners. In *Financial Cryptography and Data Security: FC 2018 International Workshops*, pages 3–18. Springer, 2019.

[72] Michael Mirkin, Yan Ji, Jonathan Pang, Ariah Klages-Mundt, Ittay Eyal, and Ari Juels. BDoS: Blockchain denial-of-service. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 601–619, 2020.

[73] Tejaswi Nadahalli, Majid Khabbazian, and Roger Wattenhofer. Timelocked bribing. In *Financial Cryptography and Data Security*, pages 53–72. Springer, 2021.

[74] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, 2008.

[75] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and cryptocurrency technologies: A comprehensive introduction*. Princeton University Press, 2016.

[76] Kevin Alarcón Negy, Peter R. Rizun, and Emin Gün Sirer. Selfish mining re-examined. In *International Conference on Financial Cryptography and Data Security*, pages 61–78. Springer, 2020.

[77] Joachim Neu, Ertem Nusret Tas, and David Tse. Ebb-and-flow protocols: A resolution of the availability-finality dilemma. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 446–465. IEEE, 2021.

[78] Joachim Neu, Ertem Nusret Tas, and David Tse. Two more attacks on proof-of-stake GHOST/Ethereum. In *Proceedings of the 2022 ACM Workshop on Developments in Consensus*, pages 43–52, 2022.

[79] Michael Neuder, Daniel J. Moroz, Rithvik Rao, and David C Parkes. Low-cost attacks on Ethereum 2.0 by sub-1/3 stakeholders. *arXiv preprint arXiv:2102.02247*, 2021.

[80] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 315–324, 2017.

[81] Rafael Pass and Elaine Shi. The sleepy model of consensus. In *Advances in Cryptology–ASIACRYPT 2017*, volume 10625, pages 380–409. Springer, 2017.

[82] Moritz Platt and Peter McBurney. Sybil in the haystack: A comprehensive review of blockchain consensus mechanisms in search of strong Sybil attack resistance. *Algorithms*, 16(1):34, 2023.

[83] Ling Ren. Analysis of Nakamoto consensus. Cryptology ePrint Archive, Paper 2019/943, 2019.

[84] Peter R. Rizun. Subchains: A technique to scale Bitcoin and improve the user experience. *Ledger*, 1:38–52, 2016.

[85] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in Bitcoin. In *Financial Cryptography and Data Security*, pages 515–532. Springer, 2017.

[86] Roozbeh Sarenche, Alireza Aghabagherloo, Svetla Nikova, and Bart Preneel. Bitcoin under volatile block rewards: How mempool statistics can influence Bitcoin mining. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, 2025. To appear, available as preprint at https://arxiv.org/abs/2411.11702.

[87] Roozbeh Sarenche, Svetla Nikova, and Bart Preneel. Deep selfish proposing in longest-chain proof-of-stake protocols. In *International Conference on Financial Cryptography and Data Security*, volume 14744, pages 24–40. Springer, 2024.

[88] Roozbeh Sarenche, Svetla Nikova, and Bart Preneel. Mining power destruction attacks in the presence of petty-compliant mining pools. In *International Conference on Financial Cryptography and Data Security*, 2025. To appear, available as preprint at https://arxiv.org/abs/2502.07410.

[89] Roozbeh Sarenche, Ertem Nusret Tas, Barnabé Monnot, Caspar Schwarz-Schilling, and Bart Preneel. Commitment attacks on Ethereum's reward mechanism. In *2025 IEEE 10th European Symposium on Security and Privacy (EuroSP)*, pages 504–526. IEEE, 2025.

[90] Roozbeh Sarenche, Ren Zhang, Svetla Nikova, and Bart Preneel. Selfish mining time-averaged analysis in Bitcoin: Is orphan reporting an effective countermeasure? *IEEE Transactions on Information Forensics and Security*, 20:449–464, 2024.

[91] Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys (CSUR)*, 22(4):299–319, 1990.

[92] Caspar Schwarz-Schilling, Joachim Neu, Barnabé Monnot, Aditya Asgaonkar, Ertem Nusret Tas, and David Tse. Three attacks on proof-of-stake Ethereum. In *International Conference on Financial Cryptography and Data Security*, pages 560–576. Springer, 2022.

[93] Elaine Shi. *Foundations of distributed consensus and blockchains*. 2020. Available at https://www.distributedconsensus.net.

[94] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT Press, 2 edition, 2018.

[95] Jason Teutsch, Sanjay Jain, and Prateek Saxena. When cryptocurrencies mine their own business. In *International Conference on Financial Cryptography and Data Security*, pages 499–514. Springer, 2016.

[96] Itay Tsabary, Matan Yechieli, Alex Manuskin, and Ittay Eyal. MAD-HTLC: because HTLC is crazy-cheap to attack. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1230–1248. IEEE, 2021.

[97] Gavin Wood. Polkadot: Vision for a heterogeneous multi-chain framework. https://polkadot.com/papers/Polkadot-whitepaper.pdf, 2016.

[98] Anatoly Yakovenko. Solana: A new architecture for a high performance blockchain. https://solana.com/solana-whitepaper.pdf, 2018.

[99] Ren Zhang and Bart Preneel. Lay down the common metrics: Evaluating proof-of-work consensus protocols' security. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 175–192. IEEE, 2019.

[100] Ren Zhang, Dingwei Zhang, Quake Wang, Shichen Wu, Jan Xie, and Bart Preneel. NC-max: Breaking the security-performance tradeoff in Nakamoto consensus. In *Proceedings 2022 Network and Distributed System Security Symposium*, pages 1–18. NDSS, 2022.

[101] Roi Bar Zur, Ittay Eyal, and Aviv Tamar. Efficient MDP analysis for selfish-mining in blockchains. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 113–131, 2020.

Part II

Publications

Chapter 6

Selfish Mining Time-Averaged Analysis in Bitcoin: Is Orphan Reporting an Effective Countermeasure?

Publication Data

Roozbeh Sarenche, Ren Zhang, Svetla Nikova, and Bart Preneel, "Selfish Mining Time-Averaged Analysis in Bitcoin: Is Orphan Reporting an Effective Countermeasure?," *IEEE Transactions on Information Forensics and Security*, volume 20, pages 449–464, 2024 (https://doi.org/10.1109/TIFS.2024.3518090).

Contributions

I was the main author involved in Conceptualization, Methodology, Writing Original Draft, Software, and Visualization.

Abstract. A Bitcoin miner who owns a sufficient amount of mining power can perform selfish mining to increase its relative revenue. Studies have demonstrated that the time-averaged profit of a selfish miner starts to rise once the mining difficulty level gets adjusted in favor of the attacker. Selfish mining profitability lies in the fact that orphan blocks are not incorporated into the current version of Bitcoin's difficulty adjustment mechanism (DAM). Therefore, it is believed that considering the count of orphan blocks in the DAM can result in complete unprofitability for selfish mining. In this paper, we disprove this belief by providing a formal analysis of the selfish mining time-averaged profit. We present a precise definition of the orphan blocks that can be incorporated into calculating the next epoch's target and then introduce two modified versions of DAM in which both main-chain blocks and orphan blocks are incorporated. We propose two versions of smart intermittent selfish mining, where the first one dominates the normal intermittent selfish mining, and the second one results in selfish mining profitability under the modified DAMs. Moreover, we present the orphan exclusion attack with the help of which the attacker can stop honest miners from reporting the orphan blocks. Using combinatorial tools, we analyze the profitability of selfish mining accompanied by the orphan exclusion attack under the modified DAMs. Our results show that even when considering orphan blocks in the DAM, selfish mining can still be profitable. However, the level of profitability under the modified DAMs is significantly lower than that observed under the current version of Bitcoin DAM, suggesting that orphan reporting can be an effective countermeasure against a payoff-maximizing selfish miner.

6.1 Introduction

One of the key challenges in designing blockchain networks is developing the underlying consensus mechanism, which ensures that all users agree on a unified ledger without the need for a central authority [40]. Bitcoin [27] employs a Proof-of-Work (PoW) mechanism to achieve consensus, where miners solve cryptographic puzzles to add new blocks to the blockchain. The first miner to solve the puzzle adds the block and receives a cryptocurrency reward. In Bitcoin, this puzzle requires finding a nonce that makes the block's hash fall below a specified difficulty target [12]. A *difficulty adjustment mechanism* (DAM) recalculates this target every 2016 blocks to maintain consistent transaction throughput. It was initially assumed that if more than half the mining power in Bitcoin follows the honest protocol, rewards would be distributed proportionally to miners' computing power [41]. However, the selfish mining attack, introduced by Eyal and Sirer in 2014, challenged this belief [9]. They demonstrated that an attacker with over 25% of the total mining power can increase its *relative revenue*, i.e., the ratio of its reward to the total distributed reward, by conducting

selfish mining. In selfish mining, the attacker withholds newly mined blocks and continues mining privately. This causes honest miners to waste their mining power on a shorter public chain. As a result, some honest blocks become orphaned, i.e., excluded from the canonical chain, enabling the attacker to earn more profit than they would by mining honestly.

In 2018, Grunspan and Pérez-Marco presented the temporal analysis of selfish mining [17]. The authors of the paper argued that the previous selfish mining papers used the Markov model [13] to analyze the selfish mining attack and thus ignored the time considerations in their analysis. More precisely, they mentioned that relative revenue is not an adequate benchmark for assessing the attacker's profitability. To properly analyze profitability, one should consider the attacker's profit per unit of time, which we refer to as *time-averaged profit*. The authors in [17] demonstrated that before a DAM, no mining strategy is more profitable than honest mining. The selfish mining attack must wait for the difficulty adjustment at the end of the attack's initial epoch to reduce the difficulty. Once the difficulty is adjusted, the time-averaged profit of selfish mining starts to rise. The authors in [17] concluded that selfish mining exploits Bitcoin's current DAM and suggested incorporating the count of orphan blocks in the DAM to mitigate the attack.

Selfish mining is often considered impractical because it is believed that the attack must be continued for a couple of epochs to become profitable. To challenge this belief, the authors in [29] introduced the intermittent selfish mining strategy, where the selfish miner alternates between selfish and honest mining at the end of each epoch. They showed that by conducting selfish mining for just one epoch and then switching to honest mining for the next epoch, a selfish miner can achieve a higher time-averaged profit over two epochs compared to mining honestly in both. The concept of intermittent selfish mining is similar to smart mining [16], where the attacker switches between honest mining and idling. Despite introducing the intermittent selfish mining attack, the paper [29] lacks a formal analysis of the attack. To address this issue, the paper [18] provided a formal analysis of intermittent selfish mining and introduced the concept of *profit lag*, defined as the earliest point after which a strategy becomes consistently profitable. The authors in [18] analyzed the profit lag for several mining strategies and demonstrated that the profit lag of intermittent selfish mining is significantly higher than that of consistent selfish mining. The intuition behind the long profit lag in intermittent selfish mining is that, although the attacker's profit at the end of an even epoch can surpass what they would have earned by following the honest strategy, the attack enters another loss period when the attacker returns to selfish mining in the subsequent odd epoch. Therefore, the profit from intermittent selfish mining, compared to honest mining, may fluctuate several times before eventually reaching a point where the attack becomes definitely profitable. The profit lag analysis of

intermittent selfish mining shows that although this attack aims to reduce the initial loss period of selfish mining, it actually prolongs the loss period.

Knowing that selfish mining can raise the attacker's relative revenue, this paper takes a closer look at the time-averaged profit of selfish mining. Although the authors in [17] have suggested incorporating the count of orphan blocks in DAM to mitigate selfish mining, the precise definition of the orphan blocks in Bitcoin needs yet to be addressed. While it is believed that orphan inclusion in DAM can make selfish mining non-profitable, in this paper, we challenge this belief by introducing two attacks: smart intermittent selfish mining and the orphan exclusion attack. These two attacks demonstrate that selfish mining can still be profitable even when orphan blocks are included in the DAM. Our contributions include:

Smart intermittent selfish mining (version 1): In Section 6.5.2, we introduce the smart intermittent selfish mining attack (version 1), which enjoys a lower profit lag and achieves higher profitability compared to the standard intermittent selfish mining introduced in [29]. In smart intermittent selfish mining (version 1), instead of alternating between full epochs of selfish mining and honest mining, the attacker divides each epoch into two parts: one for selfish mining and the other for honest mining. This approach allows the attacker to stabilize mining difficulty, which our analysis shows can enhance the attack's profitability.

Precise definition of orphan blocks: Authors in [17] introduced a new difficulty adjustment mechanism to incorporate orphaned blocks; however, they did not present a formal definition for the orphan blocks. In Section 6.6.1, we introduce uncle blocks—orphan blocks in the same epoch—to specify the properties of the valid orphan blocks that can be incorporated in the difficulty adjustment mechanism. After defining the uncle blocks, in Sections 6.6.2 and 6.6.3, we present two modified versions of the difficulty adjustment mechanism (the modified DAM) for Bitcoin in which, in addition to the main-chain blocks, the count of uncle blocks affects the mining difficulty of the next epoch.

Smart intermittent selfish mining (version 2): In Section 6.7, we introduce the smart intermittent selfish mining attack (version 2) that disproves the belief that incorporating the orphan blocks in the DAM can result in the unprofitability of selfish mining.

Orphan exclusion attack: In Section 6.8, we introduce the orphan exclusion attack with the help of which the attacker can prevent the honest miners from reporting the orphan blocks in the main chain. We show that selfish mining accompanied by the orphan exclusion attack can be profitable under the modified DAMs.

Comparison of selfish mining countermeasures: In Section 6.10, we

present a comprehensive discussion of existing selfish mining countermeasures, including the orphan reporting method.

6.2 Related Work

In this section, we provide a brief overview of the literature on the profitability of selfish mining and methods for mitigating this attack in longest-chain-based blockchains.

The concept of selfish mining was first introduced in [2], with the first theoretical analysis provided by Eyal and Sirer [9]. Research on selfish mining profitability generally follows two main directions: 1) analyzing the *block ratio* (relative revenue) achieved through selfish mining, and 2) analyzing the *time-averaged profit* of selfish mining. To analyze the block ratio, [35] proposed an MDP-based approach with a non-linear optimization function to derive the optimal selfish mining strategy, which was later refined by [44] with a linear MDP approach. A reinforcement learning tool to examine selfish mining in more realistic environments was introduced in [39]. The authors in [6, 11, 36] discussed how selfish mining also threatens longest-chain-based Proof-of-Stake (PoS) protocols, where factors such as proposer predictability and the nothing-at-stake phenomenon make the attack even more destructive. A deep Q-learning tool for analyzing selfish mining in PoS contexts was proposed in [36]. Several studies have analyzed the time-averaged profitability of selfish mining. [17] showed that although selfish mining increases relative revenue, it does not become profitable before difficulty adjustments. [29] introduced intermittent selfish mining to reduce the initial loss period associated with selfish mining. [18] introduced the concept of profit lag, the time it takes for strategies to become consistently profitable, and demonstrated that intermittent selfish mining has a longer profit lag compared to consistent selfish mining.

Various countermeasures have been proposed to mitigate selfish mining. One group of these countermeasures relies on time metrics to detect withheld blocks. [19] introduced the freshness-preferred scheme, where blocks use unforgeable timestamps to detect withheld blocks, allowing miners to choose the freshest block during fork races. Transaction creation times were employed in [23], and expected confirmation heights of transactions were proposed in [34] to identify withheld blocks. Zero-blocks, where miners create dummy blocks if no block is mined within a given time interval, were introduced in [38] and expanded with alarming blocks and block interval times in [32]. [41] proposed a fork choice rule where chain weight is determined by in-time blocks and uncles, forcing selfish miners to either publish quickly or risk losing the fork race. Other countermeasures, such as Fruitchain [31], redefine reward distribution.

In Fruitchain, miners are rewarded for *fruits*, bundles of transactions that point to earlier blocks and are less vulnerable to being orphaned by selfish miners. Another approach to limit selfish mining profitability focuses on modifying the difficulty adjustment mechanism. [43] introduced a dynamic DAM that increases difficulty when selfish mining is detected. [4] proposed a DAM sensitive to both the previous difficulty level and the estimation of current active mining power, slowing difficulty reductions caused by selfish mining. In [17], the authors suggested incorporating the number of orphaned blocks into the DAM to counter selfish mining.

For further details on selfish mining defenses, readers are referred to [25, 30, 42].

6.3 Preliminaries

In this section, we first present our system model. Then, we define the concepts of relative revenue and time-averaged profit. Finally, we discuss the effect of the difficulty adjustment mechanism on selfish mining profitability.

6.3.1 System Model and Definitions

In this paper, we use the system model introduced in [26]. We assume the system comprises a set of honest miners denoted by $\mathcal{H}$ and an adversarial miner denoted by $\mathcal{A}$. We denote by $\alpha_{\mathcal{H}}$ and $\alpha_{\mathcal{A}}$ the total honest mining power share and the adversarial mining power share, respectively, where $\alpha_{\mathcal{A}} + \alpha_{\mathcal{H}} = 1$. In our model, time is divided into rounds denoted by r. In each round, a miner can calculate multiple mining (hash) queries, the number of which is proportional to his mining power. We assume our system model is synchronous, i.e., the block published by one of the miners in round r will be delivered to all the other miners at the end of round r.

Communication capability: We denote by $\gamma_{\mathcal{A}}$ the communication capability of attacker $\mathcal{A}$. This means, in the case of a block race, where two blocks are published simultaneously by attacker $\mathcal{A}$ and an honest miner, the fraction of total honest miners that receive the block proposed by the attacker first is equal to $\gamma_{\mathcal{A}}$. Honest miners who receive the attacker's block first will mine on top of it.

The honest miners follow the honest strategy $\pi^{\mathtt{H}}$, which is explained as follows:

Honest strategy: At the start of a new round, a miner chooses to mine on top of the longest chain available in his view. If the miner manages to mine a new block, he immediately publishes the block to all the other miners.

Attacker $\mathcal{A}$ may, however, deviate from the honest strategy and mine in a selfish way. In recent years, different selfish mining strategies have been presented

such as Eyal and Sirer's selfish mining strategy $\pi^{\texttt{SM1}}$ introduced in [9], the optimal selfish mining strategy $\pi^{\texttt{OSM}}$ introduced in [35], and the intermittent selfish mining strategy $\pi^{\texttt{ISM}}$ introduced in [29]. Strategies $\pi^{\texttt{SM1}}$ and $\pi^{\texttt{OSM}}$ are specifically designed to increase a miner's relative revenue, while strategy $\pi^{\texttt{ISM}}$ aims to increase a miner's time-averaged profit. A summary of selfish mining strategies $\pi^{\texttt{SM1}}$ and $\pi^{\texttt{OSM}}$ is presented in Appendix A of [37].[1] Note that when referring to the selfish mining attack in a general context, we represent it using the notation $\pi^{\texttt{SM}}$.

Definition 6.3.1 (Relative revenue)**.** The relative revenue of attacker $\mathcal{A}$ following strategy π is defined as follows:

$$\begin{aligned} \texttt{RelRev}_{\mathcal{A}}(r;\pi) &= \frac{N^r_{\mathcal{A}}}{\sum_{\mathcal{M}\in\{\mathcal{A},\mathcal{H}\}} N^r_{\mathcal{M}}}, \quad \text{and} \\ \texttt{RelRev}_{\mathcal{A}}(\pi) &= \lim_{r\to\infty} \texttt{RelRev}_{\mathcal{A}}(r;\pi), \end{aligned} \tag{6.1}$$

where $N^r_{\mathcal{M}}$ for $\mathcal{M} \in \{\mathcal{A}, \mathcal{H}\}$ denotes the number of blocks added to the main chain by miner $\mathcal{M}$ during interval $[1, r]$.

We model block mining as a Poisson process, where the block interval time (the time between two consecutive mined blocks) follows an exponential distribution with rate parameter λ. Note $\frac{1}{\lambda}$ represents the average number of rounds it takes for the whole system to mine a new block. We denote by $R_{\mathcal{A}}(r)$ and $C_{\mathcal{A}}(r)$ the revenue and the mining cost of attacker $\mathcal{A}$ in round r, respectively. As demonstrated in [26], if all the miners including attacker $\mathcal{A}$ follow the honest strategy, the average per-round revenue of attacker $\mathcal{A}$ can be obtained as follows:

$$\mathbb{E}[R_{\mathcal{A}}(r)] = \alpha_{\mathcal{A}} \cdot \lambda K \ , \tag{6.2}$$

where K denotes the value of the mining reward per block. As outlined in [26], we assume that if attacker $\mathcal{A}$ mines with his whole mining power, his average mining cost per round can be obtained as follows:

$$\mathbb{E}[C_{\mathcal{A}}(r)] = \alpha_{\mathcal{A}} \cdot c_{\mathcal{A}} \ , \tag{6.3}$$

where $c_{\mathcal{A}}$ denotes the average normalized mining cost of miner $\mathcal{A}$ per round. Note that to compare the profitability of a strategy π with honest mining, we can disregard mining costs if, in strategy π, the attacker utilizes its entire mining power with no idle portion. In such cases, the costs are identical for both honest mining and strategy π, making the results independent of the cost formula assumption.

[1] A full version of this manuscript that includes all the definitions and proofs is presented in preprint [37].

The profitability factor of attacker $\mathcal{A}$ is denoted by $\omega_{\mathcal{A}}$ and defined as follows:

$$\omega_{\mathcal{A}} = \frac{\mathbb{E}[R_{\mathcal{A}}(r)]}{\mathbb{E}[C_{\mathcal{A}}(r)]} = \frac{\lambda K}{c_{\mathcal{A}}} \ . \tag{6.4}$$

The profitability factor $\omega_{\mathcal{A}}$ represents the amount of return per each unit of money invested by attacker $\mathcal{A}$ provided that all the miners follow the honest strategy.

Definition 6.3.2 (Time-averaged profit)**.** The time-averaged profit (per-round profit) of attacker $\mathcal{A}$ following strategy π is defined as follows:

$$\begin{aligned} \texttt{Profit}_{\mathcal{A}}(r;\pi) &= \frac{\sum_{r'=1}^{r}\Big(R_{\mathcal{A}}(r') - C_{\mathcal{A}}(r')\Big)}{r}, \quad \text{and} \\ \texttt{Profit}_{\mathcal{A}}(\pi) &= \lim_{r\to\infty} \texttt{Profit}_{\mathcal{A}}(r;\pi) \ . \end{aligned} \tag{6.5}$$

If assuming that time is divided into a set of round intervals denoted by `cycle`, according to the renewal reward process theorem, the time-averaged profit defined in Definition 6.3.2 can be obtained as follows:

$$\texttt{Profit}_{\mathcal{A}}(\pi) = \frac{\mathbb{E}\big[R_{\mathcal{A}}(\texttt{cycle})\big] - \mathbb{E}\big[C_{\mathcal{A}}(\texttt{cycle})\big]}{\mathbb{E}[t(\texttt{cycle})]} \ , \tag{6.6}$$

where $t(\texttt{cycle})$ represents the duration of `cycle`, and $R_{\mathcal{A}}(\texttt{cycle})$ and $C_{\mathcal{A}}(\texttt{cycle})$ denote the revenue and the mining cost of attacker $\mathcal{A}$ within the `cycle`, respectively.

6.4 Background on Selfish Mining Profitability

Unprofitability of selfish mining before a difficulty adjustment mechanism In Bitcoin, an interval of rounds in which a set of $L = 2016$ consecutive blocks is added to the main chain is called an epoch. At the end of each epoch, there is a difficulty adjustment mechanism (DAM), which calculates the difficulty target of the upcoming epoch based on the hash power estimation of the previous epoch. Assume attacker $\mathcal{A}$ starts the selfish mining attack at the beginning of $\texttt{epoch}_1$. As it is discussed in [17] and [29], the time-averaged profit of attacker $\mathcal{A}$ under selfish mining cannot exceed his time-averaged profit under the honest strategy during $\texttt{epoch}_1$, i.e., before the next DAM. To illustrate this fact, we need to compare the attacker's profitability when following the honest strategy versus the selfish strategy. Note that the mining difficulty of $\texttt{epoch}_1$ is specified before the start of $\texttt{epoch}_1$, and thus, the attacker's strategy during $\texttt{epoch}_1$ cannot

change the epoch's mining difficulty. If attacker $\mathcal{A}$ follows the honest strategy in $\texttt{epoch}_1$, he can mine a new block every $\frac{1}{\lambda\alpha_{\mathcal{A}}}$ rounds on average, and if ignoring the natural orphan occurrence, all of his blocks will be added to the main chain. If attacker $\mathcal{A}$ performs the selfish mining attack during $\texttt{epoch}_1$, since the mining difficulty is the same as the former scenario, his average mining rate is still equal to $\frac{1}{\lambda\alpha_{\mathcal{A}}}$; however, in this scenario, some of the attacker's blocks may get orphaned and remain out of the main chain due to the block races caused by the selfish mining attack. Therefore, before a DAM, the time-averaged profit of selfish mining cannot exceed the time-averaged profit of honest mining. Note that selfish mining can potentially increase the attacker's relative revenue in $\texttt{epoch}_1$. However, despite this increase in relative revenue, the attacker cannot gain a higher time-averaged profit during the first epoch of the attack.

The effect of DAM on selfish mining profitability In Bitcoin, DAM is responsible for adjusting the block generation rate to ensure that, on average, it takes 10 minutes for the system to mine a new block. Therefore, the average epoch duration is equal to 2 weeks. Let $\texttt{epoch}_1$ and $\texttt{epoch}_2$ denote two consecutive epochs. If attacker $\mathcal{A}$ starts selfish mining in $\texttt{epoch}_1$, some of the both honest and adversarial blocks will get orphaned in $\texttt{epoch}_1$. Consequently, the duration in which L blocks are added to the main chain will be extended. This implies that the duration of $\texttt{epoch}_1$ will increase, exceeding the standard two-week period. At the end of $\texttt{epoch}_1$, there is a DAM that calculates the mining difficulty of $\texttt{epoch}_2$ based on the hash power estimation of $\texttt{epoch}_1$. Since the current version of Bitcoin DAM does not consider orphan blocks when estimating the active hash power of the previous epoch, the increase in the length of $\texttt{epoch}_1$ will result in a decrease in the mining difficulty of $\texttt{epoch}_2$. Therefore, during $\texttt{epoch}_2$, attacker $\mathcal{A}$ can mine a new block, on average, within a shorter period than $\frac{1}{\lambda\alpha_{\mathcal{A}}}$ rounds. This shows that starting from $\texttt{epoch}_2$, the attacker's time-averaged profit begins to increase.

Selfish mining profitability after the adjustment of mining difficulty Assume attacker $\mathcal{A}$ has started selfish mining in $\texttt{epoch}_1$. Therefore, the mining difficulty of the next epoch, i.e., $\texttt{epoch}_2$, is adjusted in favor of the attacker. The time-averaged profit of attacker $\mathcal{A}$ in $\texttt{epoch}_2$ under the selfish mining strategy can be obtained as follows:

$$\texttt{Profit}_{\mathcal{A}}(\pi^{\texttt{SM}}) = \lambda K \cdot \texttt{RelRev}_{\mathcal{A}}(\pi^{\texttt{SM}}) - \alpha_{\mathcal{A}} c_{\mathcal{A}} \ . \tag{6.7}$$

Note that the time-averaged profit of attacker $\mathcal{A}$ under the honest strategy is equal to:

$$\texttt{Profit}_{\mathcal{A}}(\pi^{\texttt{H}}) = \alpha_{\mathcal{A}}(\lambda K - c_{\mathcal{A}}) \ . \tag{6.8}$$

This shows that if the attacker's selfish mining relative revenue is greater than his honest mining relative revenue, i.e., $\texttt{RelRev}_{\mathcal{A}}(\pi^{\texttt{SM}}) > \alpha_{\mathcal{A}}$, the selfish mining strategy dominates the honest strategy after the adjustment of mining difficulty. It has been shown that attacker $\mathcal{A}$ with a normal communication capability $\gamma_{\mathcal{A}} = 0.5$ needs to own more than 25% of the total mining power to achieve $\texttt{RelRev}_{\mathcal{A}}(\pi^{\texttt{SM}}) > \alpha_{\mathcal{A}}$ [9].

Knowing that selfish mining profit can surpass honest mining profit, a question arises as to why miners are unwilling to perform the selfish mining attack. One of the main reasons that answers this question is the belief that an attacker should perform selfish mining for a relatively long time to gain profit. In other words, selfish mining always begins with an initial period of loss. Once the attacker starts selfish mining in $\texttt{epoch}_1$, since some of the attacker's blocks get orphaned, his gained profit in $\texttt{epoch}_1$ will be lower than his profit under honest mining. Therefore, to consider selfish mining as a profitable strategy, the gained profit by the attacker in $\texttt{epoch}_2$ (or even later epochs) should compensate for the loss the attacker has faced in $\texttt{epoch}_1$. However, if the attacker continues selfish mining for a considerable number of epochs, honest miners may decide to stop mining to avoid financial losses.

Intermittent selfish mining and profit lag The authors in [29] introduced the intermittent selfish mining (ISM) attack to reduce the initial loss period of selfish mining. In ISM strategy, the attacker alternates between selfish mining and honest mining at every DAM. In other words, for two consecutive epochs denoted by $\texttt{epoch}_1$ and $\texttt{epoch}_2$, the attacker applies the selfish mining attack in $\texttt{epoch}_1$ and returns to honest mining in $\texttt{epoch}_2$. As already discussed, by selfish mining in $\texttt{epoch}_1$, the attacker cannot increase his time-averaged profit immediately in $\texttt{epoch}_1$; however, the selfish mining attack in $\texttt{epoch}_1$ can lead to a decrease in the mining difficulty of $\texttt{epoch}_2$. To take full advantage of the decreased mining difficulty in $\texttt{epoch}_2$, the authors in [29] suggested that, instead of continuing selfish mining, the attacker should mine honestly in $\texttt{epoch}_2$ to collect all the possible blocks and maximize its rewards. The authors in [29] concluded that intermittent selfish mining can shorten the attack's loss period. However, the authors in [18] revisited this conclusion, emphasizing that the profit achieved by following a mining strategy can fluctuate multiple times before definitively surpassing honest mining profitability. They argued that while ISM profit at the end of the second epoch may exceed that of honest mining, once the third epoch begins and the attacker returns to selfish mining, the profitability of the attack again falls below what could have been achieved through honest mining over the previous three epochs. To better analyze the initial loss period of a mining strategy, the authors in [18] introduced the concept of profit lag. The profit lag of a mining strategy π is defined as the smallest time point τ, where for any $t > \tau$, the following property holds for the attacker's

time-averaged profit: $\mathbb{E}\big[\texttt{Profit}_{\mathcal{A}}(t;\pi)\big] \geq \mathbb{E}\big[\texttt{Profit}_{\mathcal{A}}(t;\pi^{\texttt{H}})\big]$. Intuitively, the profit lag is the smallest time since the start of the attack after which the attacker's time-averaged profit consistently surpasses that of honest mining. The authors in [18] demonstrated that the profit lag of intermittent selfish mining is significantly longer than that of consistent selfish mining, implying that intermittent selfish mining cannot effectively reduce the initial loss period of selfish mining.

6.5 Intermittent Selfish Mining

Despite introducing intermittent selfish mining (ISM) in [29], the authors did not provide a formal analysis. The authors in [18] formalized the analysis of ISM based on the selfish mining strategy from [9]. We extend this analysis in Section 6.5.1 by considering the optimal selfish mining strategy introduced in [35]. Besides, in Section 6.5.2, we introduce the smart intermittent selfish mining attack.

6.5.1 Normal Intermittent Selfish Mining

We denote by $\pi^{\texttt{ISM}}$ the ISM strategy. In the ISM strategy, the attacker performs selfish mining in odd epochs, i.e., $\{\texttt{epoch}_1, \texttt{epoch}_3, \cdots\}$, and applies honest mining in even epochs, i.e., $\{\texttt{epoch}_2, \texttt{epoch}_4, \cdots\}$. Using equation 6.6, the attacker's time-averaged profit under the intermittent selfish mining can be obtained as follows:

$$\begin{aligned}&\texttt{Profit}_{\mathcal{A}}(\pi^{\texttt{ISM}}) = \\ &\frac{\mathbb{E}\big[R_{\mathcal{A}}(\texttt{epoch}_{\texttt{odd}})\big] - \mathbb{E}\big[C_{\mathcal{A}}(\texttt{epoch}_{\texttt{odd}})\big]}{\mathbb{E}[t(\texttt{epoch}_{\texttt{odd}})] + \mathbb{E}[t(\texttt{epoch}_{\texttt{even}})]} + \\ &\frac{\mathbb{E}\big[R_{\mathcal{A}}(\texttt{epoch}_{\texttt{even}})\big] - \mathbb{E}\big[C_{\mathcal{A}}(\texttt{epoch}_{\texttt{even}})\big]}{\mathbb{E}[t(\texttt{epoch}_{\texttt{odd}})] + \mathbb{E}[t(\texttt{epoch}_{\texttt{even}})]} \ .\end{aligned} \tag{6.9}$$

For simplicity, we refrain from using the expected value notation, denoted as $\mathbb{E}[\cdot]$, throughout the rest of the paper. The average revenue gained by the attacker in $\texttt{epoch}_{\texttt{odd}}$ and his mining cost in $\texttt{epoch}_{\texttt{odd}}$ can be obtained as follows:

$$\begin{aligned}R_{\mathcal{A}}(\texttt{epoch}_{\texttt{odd}}) &= \texttt{RelRev}\big(\pi^{\texttt{SM}}\big) \cdot LK \ , \\ C_{\mathcal{A}}(\texttt{epoch}_{\texttt{odd}}) &= \alpha_{\mathcal{A}} c_{\mathcal{A}} \cdot t(\texttt{epoch}_{\texttt{odd}}) \ .\end{aligned} \tag{6.10}$$

Similarly, the average revenue gained by the attacker in $\texttt{epoch}_{\texttt{even}}$ and his mining cost in $\texttt{epoch}_{\texttt{even}}$ can be obtained as follows:

$$\begin{aligned} R_{\mathcal{A}}(\texttt{epoch}_{\texttt{even}}) &= \alpha_{\mathcal{A}} \cdot LK \ , \\ C_{\mathcal{A}}(\texttt{epoch}_{\texttt{even}}) &= \alpha_{\mathcal{A}} c_{\mathcal{A}} \cdot t(\texttt{epoch}_{\texttt{even}}) \ . \end{aligned} \tag{6.11}$$

To obtain the epoch duration, we first define the concepts of active and effective mining power.

Definition 6.5.1 (Active and effective mining power)**.** The normalized active mining power of an epoch is defined as the ratio of mining power dedicated to mining both orphan and main-chain blocks during that epoch to the total available mining power. The normalized effective mining power of an epoch is defined as the ratio of mining power dedicated solely to mining main-chain blocks during that epoch to the total available mining power.

We denote by $M_{\texttt{odd}}^{\texttt{total}}$ and $M_{\texttt{odd}}^{\texttt{main-chain}}$ the normalized active mining power in $\texttt{epoch}_{\texttt{odd}}$ and the normalized effective mining power extending the main chain in $\texttt{epoch}_{\texttt{odd}}$, respectively. As there is no idle power in $\texttt{epoch}_{\texttt{odd}}$, we have $M_{\texttt{odd}}^{\texttt{total}} = 1$. The effective mining power $M_{\texttt{odd}}^{\texttt{main-chain}}$ represents the ratio of the number of main-chain blocks in $\texttt{epoch}_{\texttt{odd}}$ to the total number of blocks mined during $\texttt{epoch}_{\texttt{odd}}$. Due to the selfish mining attack in $\texttt{epoch}_{\texttt{odd}}$ and orphan occurrence, some part of the mining power in $\texttt{epoch}_{\texttt{odd}}$ gets wasted and does not contribute to extending the main chain. This implies that $M_{\texttt{odd}}^{\texttt{main-chain}}$ is less than 1. We define $M_{\texttt{SM}}^{\texttt{main-chain}}$ to be the normalized effective mining power under selfish mining. We have $M_{\texttt{odd}}^{\texttt{main-chain}} = M_{\texttt{SM}}^{\texttt{main-chain}}$. In Appendix A of [37], the methods for calculating $M_{\texttt{SM}}^{\texttt{main-chain}}$ under both selfish mining strategies $\pi^{\texttt{SM1}}$ and $\pi^{\texttt{OSM}}$ are explained. Similar terms can be defined for $\texttt{epoch}_{\texttt{even}}$. Since all the miners follow the honest strategy in $\texttt{epoch}_{\texttt{even}}$, we have $M_{\texttt{even}}^{\texttt{main-chain}} = M_{\texttt{even}}^{\texttt{total}} = 1$. According to the design of the current version of Bitcoin's DAM, the duration of $\texttt{epoch}_{\texttt{odd}}$ can be calculated as follows:

$$t(\texttt{epoch}_{\texttt{odd}}) = t^{\texttt{ideal}} \cdot \frac{M_{\texttt{even}}^{\texttt{main-chain}}}{M_{\texttt{odd}}^{\texttt{main-chain}}} = \frac{L}{\lambda M_{\texttt{SM}}^{\texttt{main-chain}}} \ , \tag{6.12}$$

where $t^{\texttt{ideal}}$ represents the ideal epoch duration and is equal to $\frac{L}{\lambda}$. Note that under the current version of Bitcoin DAM, the epoch duration $t(\texttt{epoch}_{\texttt{odd}})$ is inversely related to the epoch's main-chain effective power $M_{\texttt{odd}}^{\texttt{main-chain}}$. The greater the amount of mining power working to extend the main chain within an epoch, the shorter the time it takes for the epoch to complete. However, the epoch duration $t(\texttt{epoch}_{\texttt{odd}})$ is directly related to the previous epoch's main-chain effective power $M_{\texttt{even}}^{\texttt{main-chain}}$. The reason is that a lower amount of main-chain effective mining power in the previous epoch results in a decrease in the mining

difficulty and consequently the duration of the current epoch. Similarly, the duration of $\texttt{epoch}_{\texttt{even}}$ can be calculated as follows:

$$t(\texttt{epoch}_{\texttt{even}}) = t^{\texttt{ideal}} \cdot \frac{M_{\texttt{odd}}^{\texttt{main-chain}}}{M_{\texttt{even}}^{\texttt{main-chain}}} = \frac{LM_{\texttt{SM}}^{\texttt{main-chain}}}{\lambda} \ . \tag{6.13}$$

Therefore, the attacker's time-averaged profit under intermittent selfish mining can be obtained as follows:

$$\texttt{Profit}_{\mathcal{A}}(\pi^{\texttt{ISM}}) = \frac{\lambda K\Big(\texttt{RelRev}\big(\pi^{\texttt{SM}}\big) + \alpha_{\mathcal{A}}\Big)}{\frac{1}{M_{\texttt{SM}}^{\texttt{main-chain}}} + M_{\texttt{SM}}^{\texttt{main-chain}}} - \alpha_{\mathcal{A}} c_{\mathcal{A}} \ . \tag{6.14}$$

Mining power threshold values: We aim to address the question of how much mining power is required to make the intermittent selfish mining strategy more profitable than the honest strategy. By fixing the attacker's communication capability $\gamma_{\mathcal{A}}$, we want to calculate the minimum amount of the attacker's mining power that satisfies the following inequality:

$$\texttt{Profit}_{\mathcal{A}}(\pi^{\texttt{ISM}}) > \texttt{Profit}_{\mathcal{A}}(\pi^{\texttt{H}}) \ . \tag{6.15}$$

Using equations 6.8 and 6.14, we can obtain that to achieve the inequality above, the following inequality should hold:

$$\texttt{RelRev}\big(\pi^{\texttt{SM}}\big) > \Big(\frac{1}{M_{\texttt{SM}}^{\texttt{main-chain}}} + M_{\texttt{SM}}^{\texttt{main-chain}} - 1\Big)\alpha_{\mathcal{A}} \ . \tag{6.16}$$

As the coefficient of $\alpha_{\mathcal{A}}$ is greater than 1 in the inequality above, the minimum amount of mining power that makes ISM profitable is more than that in normal selfish mining. For instance, for a normal communication capability $\gamma_{\mathcal{A}} = 0.5$, the minimum amount of mining power share that makes ISM profitable is equal to 0.2773.

6.5.2 Smart Intermittent Selfish Mining (Version 1)

In intermittent selfish mining, the attacker performs the attack every other epoch. This shows that the ratio of the selfish mining period length (measured by the number of blocks added to the main chain) to the main chain length is equal to $\frac{1}{2}$. In the smart intermittent selfish mining attack (version 1) denoted by SISM1, the attacker gains a higher amount of profit while his ratio of selfish mining period length to the main-chain length is still equal to $\frac{1}{2}$. Assume $\texttt{epoch}_{\texttt{odd}}$ and $\texttt{epoch}_{\texttt{even}}$ are two consecutive epochs in which $2L$ blocks are added to the main chain. In SISM1, the attacker performs selfish mining for

$(1-\eta)L$ blocks in $\texttt{epoch}_{\texttt{odd}}$ and for ηL blocks in $\texttt{epoch}_{\texttt{even}}$, where $0 \leq \eta \leq 0.5$. For the remaining blocks in these two epochs, the attacker follows the honest mining strategy. It is clear that in SISM1, the ratio of selfish mining period length to the main-chain length is equal to $\frac{1}{2}$. Note that if $\eta = 0$, SISM1 is the same as the normal intermittent selfish mining attack. The duration of $\texttt{epoch}_{\texttt{odd}}$ and $\texttt{epoch}_{\texttt{even}}$ in SISM1 can be calculated as follows:

$$\begin{aligned} t(\texttt{epoch}_{\texttt{odd}}) &= t^{\texttt{ideal}} \frac{\eta + \frac{1-\eta}{M_{\texttt{SM}}^{\texttt{main-chain}}}}{1-\eta+\frac{\eta}{M_{\texttt{SM}}^{\texttt{main-chain}}}} \quad , \\ t(\texttt{epoch}_{\texttt{even}}) &= t^{\texttt{ideal}} \frac{1-\eta+\frac{\eta}{M_{\texttt{SM}}^{\texttt{main-chain}}}}{\eta+\frac{1-\eta}{M_{\texttt{SM}}^{\texttt{main-chain}}}} \quad . \end{aligned} \tag{6.17}$$

Therefore, the attacker's time-averaged profit under SISM1 is equal to:

$$\begin{aligned} &\texttt{Profit}_{\mathcal{A}}(\pi^{\texttt{SISM1}}) = \\ &\frac{\lambda K\Big(\texttt{RelRev}\big(\pi^{\texttt{SM}}\big) + \alpha_{\mathcal{A}}\Big)}{\frac{\eta+\frac{1-\eta}{M_{\texttt{SM}}^{\texttt{main-chain}}}}{1-\eta+\frac{\eta}{M_{\texttt{SM}}^{\texttt{main-chain}}}} + \frac{1-\eta+\frac{\eta}{M_{\texttt{SM}}^{\texttt{main-chain}}}}{\eta+\frac{1-\eta}{M_{\texttt{SM}}^{\texttt{main-chain}}}}} - \alpha_{\mathcal{A}} c_{\mathcal{A}} \quad . \end{aligned} \tag{6.18}$$

The maximum amount of the attacker's time-averaged profit under SISM1 occurs when $\eta = 0.5$:

$$\begin{aligned} &\texttt{Profit}_{\mathcal{A}}\big(\pi^{\texttt{SISM1}} | \eta = 0.5\big) = \\ &\frac{\lambda K\Big(\texttt{RelRev}\big(\pi^{\texttt{SM}}\big) + \alpha_{\mathcal{A}}\Big)}{2} - \alpha_{\mathcal{A}} c_{\mathcal{A}} \quad . \end{aligned} \tag{6.19}$$

Comparing equations 6.14 and 6.19, it is clear that the optimal SISM1 dominates the normal intermittent selfish mining strategy, i.e., $\texttt{Profit}_{\mathcal{A}}\big(\pi^{\texttt{SISM1}} | \eta = 0.5\big) > \texttt{Profit}_{\mathcal{A}}\big(\pi^{\texttt{ISM}}\big)$. One can consider the optimal SISM1 as the normal intermittent selfish mining with the difference that the DAM is exactly placed in the middle of the selfish mining period. Therefore, the optimal SISM1 is equivalent to performing selfish mining for half of each epoch. This shows that to increase his time-averaged profit, the attacker should keep the mining difficulty level constant and avoid inducing fluctuations. Note that the minimum amount of mining power that makes the optimal SISM1 profitable is the same as that in normal selfish mining. This implies that for a normal communication capability $\gamma_{\mathcal{A}} = 0.5$, the minimum amount of mining power share that makes SISM1 profitable is equal to 0.25.

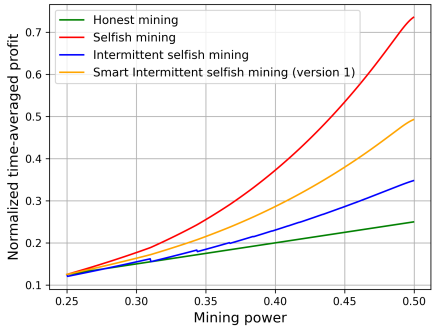


Figure 6.1: Time-averaged profit under the Bitcoin's DAM.

The normalized time-averaged profit, which is equal to $\frac{\texttt{Profit}_{\mathcal{A}}(\pi)}{\lambda K}$, is depicted in Figure 6.1 for multiple strategies. In this figure, the profitability factor $\omega_{\mathcal{A}}$ is set to 2, and the optimal selfish mining strategy $\pi^{\texttt{OSM}}$ is used to calculate $\texttt{RelRev}\big(\pi^{\texttt{SM}}\big)$ and $M_{\texttt{SM}}^{\texttt{main-chain}}$. An interesting observation regarding Figure 6.1 is that the time-averaged profit of intermittent selfish mining does not necessarily increase with the amount of mining power. The reason is that $\pi^{\texttt{OSM}}$ is specifically designed to maximize the relative revenue of selfish mining. However, $\texttt{Profit}_{\mathcal{A}}(\pi^{\texttt{ISM}})$ not only depends on $\texttt{RelRev}\big(\pi^{\texttt{SM}}\big)$, but it also depends on $M_{\texttt{SM}}^{\texttt{main-chain}}$, a point that is not considered in the design of $\pi^{\texttt{OSM}}$. This shows that the strategy optimizing the time-averaged profit of intermittent selfish mining differs from the strategy that maximizes selfish mining relative revenue.

We define the revenue advantage of strategy π as the difference between the revenues achieved by following strategy π and the honest strategy, measured from the start of employing strategy π. To compare the profit lag [18] of different selfish mining strategies, Figures 6.2 and 6.3 depict the revenue advantage of mining strategies as a function of time (in 2-week units) since the start of the attack, for an adversary with a mining share of 0.27 and $\frac{1}{3}$, respectively, both assuming a normal communication capability of 0.5. In Figures 6.2 and 6.3, the total epoch reward is normalized to 1. It is also assumed that in SISM1, the adversary dedicates the second half of each epoch to conducting the selfish mining attack. In [18], it is shown that the profit lag of consistent selfish mining is shorter than that of intermittent selfish mining. As seen in Figures 6.2 and 6.3, the SISM1 strategy achieves a shorter profit lag, i.e., the initial period of loss, compared to both consistent and intermittent selfish mining strategies. Note that when the adversary's mining share is 0.27, the intermittent selfish mining attack is unprofitable, and its revenue advantage remains consistently below zero.

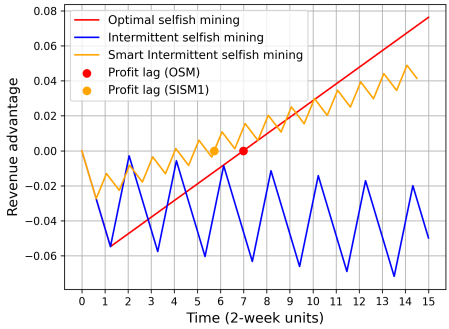


Figure 6.2: Profit lag under the Bitcoin's DAM. Adversarial mining power share and communication capability are 0.27 and 0.5, respectively.

6.6 Selfish Mining under the Modified DAM

In this section, we introduce two versions of a modified Bitcoin DAM, incorporating both main-chain and orphan blocks in the difficulty calculation. We define the orphan blocks eligible for consideration in the modified DAMs and evaluate the impact of modified DAMs on selfish mining profitability.

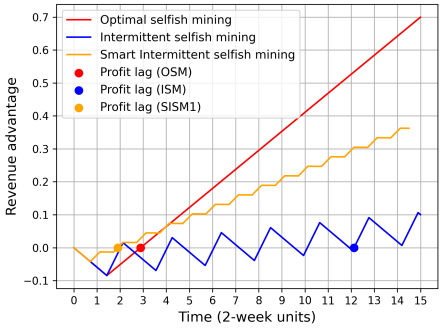


Figure 6.3: Profit lag under the Bitcoin's DAM. Adversarial mining power share and communication capability are $\frac{1}{3}$ and 0.5, respectively.

6.6.1 The Uncle Blocks

The current Bitcoin DAM is presented in Appendix B of [37]. To present the modified DAMs, we first need to define the uncle blocks. In the modified DAMs, we aim to consider the orphan blocks in the hash rate estimation. The uncles are the orphan blocks that are mined during the same epoch.

Definition 6.6.1 (Uncle)**.** A block B_1 is considered an *uncle* of another block B_2 if all of the following conditions are met: (1) they are in the same epoch, sharing the same difficulty; (2) $height(B_2) > height(B_1)$; (3) The chains containing blocks B_1 and B_2 share a common subchain, which includes at least the genesis block; and (4) B_2 is the first block in its chain to refer to B_1.

Condition (1) allows uncles to contribute to a more accurate hash rate estimation and prevents adversarial behavior such as revealing old orphan blocks to manipulate the mining difficulty. A violation of (2) contradicts the longest-chain rule. Condition (3) is to ensure that two blockchain instances with different genesis blocks do not accidentally recognize each other's blocks as uncles. Besides, B_1's parent is needed to verify the validity of B_1. Condition (4) is to prevent honest miners from reporting the same uncle multiple times.

Incorporating uncles: Miners are requested to refer to uncles—orphaned blocks in the same epoch—by embedding their hashes in the blocks. Embedding uncles contributes to a more accurate estimation of the network hash rate, thus contributing to the system's selfish mining resistance. Even a single honest block can report all previously unreported uncles in the same epoch. Note that our uncle's definition differs from that of Ethereum's deprecated PoW version [7] in that our uncle's validity does not consider how far away the uncle and the nephew's first common ancestor is. The reward distribution regarding uncles is also different from that of Ethereum. Our design issues neither uncle rewards to compensate uncle miners, nor nephew rewards to incentivize miners to embed uncles. This is because uncle and nephew rewards raise the selfish mining profit and lower the mining power threshold to perform the attack [10, 33].

In the following, two versions of the modified DAM are introduced. These two versions differ from each other in their definition of the epoch, the period at the end of which the DAM is applied. Note that our modified DAMs are nearly the same as the DAM introduced in [17], with the difference that our modified DAMs only incorporate the orphan blocks that satisfy Definition 6.6.1. In the remainder of this paper, we use the terms uncle and orphan blocks interchangeably.

6.6.2 Modified DAM with a Fixed Total Block Count per Epoch

Let L denote the length of each epoch in the current version of Bitcoin DAM, i.e., $L = 2016$. In this version of the modified DAM, denoted by $\mathsf{DAM}_1^{\texttt{modified}}$, an epoch is defined as an interval of rounds in which a set of L blocks (comprising both main-chain and orphan blocks) are mined and subsequently reported in the main chain. Let $\mathsf{CNT}_i^{\texttt{main-chain}}$, $\mathsf{CNT}_i^{\texttt{orphan}}$, and $\mathsf{CNT}_i^{\texttt{total}}$ denote the number of main-chain blocks, the number of orphan blocks embedded in the main chain, and the total number of both main-chain and reported orphan blocks mined during epoch i, respectively. Note that if assuming that all the orphan blocks are reported, the ratio $\mathsf{CNT}_i^{\texttt{main-chain}}/(\mathsf{CNT}_i^{\texttt{orphan}} + \mathsf{CNT}_i^{\texttt{main-chain}})$ represents the epoch's effective main-chain power, i.e., $M^{\texttt{main-chain}}$. According to the definition of epoch in $\mathsf{DAM}_1^{\texttt{modified}}$, we have $\mathsf{CNT}_i^{\texttt{total}} = L$ and $\mathsf{CNT}_i^{\texttt{main-chain}} \leq L$. This indicates that in $\mathsf{DAM}_1^{\texttt{modified}}$, the total number of blocks mined per epoch is fixed; however, the total amount of distributed reward per epoch can vary across different epochs.

Inputs and outputs: Similar to the Bitcoin DAM, our modified DAM is executed at the end of every epoch. It takes two inputs: the last epoch's target denoted by TGT_i and the last epoch's duration—the timestamp difference between epoch i and $i-1$'s last blocks—denoted by t_i. Note that TGT_i is decided by the last DAM iteration, and t_i is measured after the epoch ends. The algorithm outputs the next epoch's target denoted by TGT_{i+1}.

To adjust the target TGT based on the network hash rate, the modified $\mathsf{DAM}_1^{\texttt{modified}}$ is triggered at the end of each epoch as follows:

$$\mathsf{TGT}_{i+1} = \begin{cases} \mathsf{TGT}_i \cdot \frac{1}{\tau}, & t_i < \frac{1}{\tau} \cdot t_{\text{ideal}} \\ \mathsf{TGT}_i \cdot \tau, & t_i > \tau \cdot t_{\text{ideal}} \\ \mathsf{TGT}_i \cdot \frac{t_i}{t_{\text{ideal}}}, & \text{otherwise} \end{cases}, \tag{6.20}$$

where i is the epoch number, and τ is a dampening filter to prevent rapid changes of TGT. Similar to the Bitcoin DAM, we assume $t^{\texttt{ideal}}$ for $\mathsf{DAM}_1^{\texttt{modified}}$ is two weeks.

6.6.3 Modified DAM with a Fixed Main-Chain Block Count per Epoch

We use the same notations as those introduced in the $\mathsf{DAM}_1^{\texttt{modified}}$ explanation. In this version of the modified DAM, denoted by $\mathsf{DAM}_2^{\texttt{modified}}$, an epoch is defined as an interval of rounds in which a set of L consecutive blocks is added to the main chain. According to the definition of epoch in $\mathsf{DAM}_2^{\texttt{modified}}$, we

have $\mathsf{CNT}_i^{\texttt{main-chain}} = L$ and $\mathsf{CNT}_i^{\texttt{total}} \geq L$. This indicates that in $\mathsf{DAM}_2^{\texttt{modified}}$, the total amount of distributed reward per epoch is fixed; however, the total number of blocks mined per epoch may vary across different epochs.

Inputs and outputs: The modified DAM takes three inputs: the last epoch's target denoted by TGT_i, the last epoch's duration—the timestamp difference between epoch i and $i-1$'s last blocks—denoted by t_i, and the last epoch's orphaned block count—the number of uncles embedded in epoch i's main chain—denoted by $\mathsf{CNT}_i^{\texttt{orphan}}$. Among these inputs, TGT_i is decided by the last DAM iteration, while t_i and $\mathsf{CNT}_i^{\texttt{orphan}}$ are measured after the epoch ends.

To adjust the target TGT based on the network hash rate, the modified $\mathsf{DAM}_2^{\texttt{modified}}$ is triggered at the end of each epoch as follows:

$$\mathsf{TGT}_{i+1} = \begin{cases} \mathsf{TGT}_i \cdot \frac{1}{\tau}, & t_i < \frac{1}{\tau} \cdot T \\ \mathsf{TGT}_i \cdot \tau, & t_i > \tau \cdot T \\ \mathsf{TGT}_i \cdot \frac{t_i}{T}, & \text{otherwise} \end{cases}, \tag{6.21}$$

where $t^{\texttt{ideal}}$ is two weeks, and period T is defined as follows:

$$T = \frac{\mathsf{CNT}_i^{\texttt{orphan}} + \mathsf{CNT}_i^{\texttt{main-chain}}}{\mathsf{CNT}_i^{\texttt{main-chain}}} \cdot t^{\texttt{ideal}} \tag{6.22}$$

In both versions of the modified DAM, in addition to the main-chain blocks, the count of orphan blocks is considered in the calculation of the mining target. In the remainder of the paper, whenever we want to refer to a DAM that incorporates the orphan blocks, we will use the term "the modified DAM" without explicitly indicating its exact version.

6.6.4 Normal Selfish Mining under the Modified DAM

In this section, we analyze the normal selfish mining profitability under the modified DAM. By normal selfish mining attack, we mean that the attacker follows the selfish mining strategy continuously for all the epochs.

6.6.4.1 Analysis under $\mathsf{DAM}_1^{\texttt{modified}}$

We first obtain the average revenue gained by the attacker in each epoch and his mining cost. Note that due to the selfish mining attack and the definition of epoch in $\mathsf{DAM}_1^{\texttt{modified}}$, the average number of main-chain blocks in each epoch is equal to $M_{\texttt{SM}}^{\texttt{main-chain}} L$. As a result, the total amount of distributed reward

per epoch is equal to $M_{\mathtt{SM}}^{\mathtt{main\text{-}chain}} LK$.

$$\begin{aligned} R_{\mathcal{A}}(\mathtt{epoch}_i) &= \mathtt{RelRev}\big(\pi^{\mathtt{SM}}\big) \cdot M_{\mathtt{SM}}^{\mathtt{main\text{-}chain}} LK \ , \\ C_{\mathcal{A}}(\mathtt{epoch}) &= \alpha_{\mathcal{A}} c_{\mathcal{A}} \cdot t(\mathtt{epoch}_i) \ . \end{aligned} \tag{6.23}$$

Then, we calculate the epoch duration. Note that $M_{i-1}^{\mathtt{total}} = M_i^{\mathtt{total}} = 1$. According to the design of $\mathsf{DAM}_1^{\mathtt{modified}}$, the duration of each epoch can be calculated as follows.

$$t(\mathtt{epoch}_i) = t^{\mathtt{ideal}} \cdot \frac{M_{i-1}^{\mathtt{total}}}{M_i^{\mathtt{total}}} = \frac{L}{\lambda} \ . \tag{6.24}$$

Therefore, the time-averaged profit of normal selfish mining under $\mathsf{DAM}_1^{\mathtt{modified}}$ is equal to:

$$\begin{aligned} &\mathtt{Profit}_{\mathcal{A}}(\pi^{\mathtt{SM}}, \mathsf{DAM}_1^{\mathtt{modified}}) = \\ &\lambda K \Big(\mathtt{RelRev}\big(\pi^{\mathtt{SM}}\big) M_{\mathtt{SM}}^{\mathtt{main\text{-}chain}} \Big) - \alpha_{\mathcal{A}} c_{\mathcal{A}} \ . \end{aligned} \tag{6.25}$$

6.6.4.2 Analysis under $\mathsf{DAM}_2^{\mathtt{modified}}$

According to the definition of epoch in $\mathsf{DAM}_2^{\mathtt{modified}}$, the number of main-chain blocks and the total amount of distributed reward in each epoch are equal to L and LK, respectively. Therefore, the average revenue gained by the attacker in each epoch and his mining cost can be obtained as follows:

$$\begin{aligned} R_{\mathcal{A}}(\mathtt{epoch}_i) &= \mathtt{RelRev}\big(\pi^{\mathtt{SM}}\big) \cdot LK \ , \\ C_{\mathcal{A}}(\mathtt{epoch}) &= \alpha_{\mathcal{A}} c_{\mathcal{A}} \cdot t(\mathtt{epoch}_i) \ . \end{aligned} \tag{6.26}$$

To calculate the epoch duration, one should consider that $M_{i-1}^{\mathtt{total}} = 1$ and $M_i^{\mathtt{main\text{-}chain}} = M_{\mathtt{SM}}^{\mathtt{main\text{-}chain}}$. According to the design of $\mathsf{DAM}_2^{\mathtt{modified}}$, the duration of each epoch can be calculated as follows.

$$t(\mathtt{epoch}_i) = t^{\mathtt{ideal}} \cdot \frac{M_{i-1}^{\mathtt{total}}}{M_i^{\mathtt{main\text{-}chain}}} = \frac{L}{\lambda M_{\mathtt{SM}}^{\mathtt{main\text{-}chain}}} \ . \tag{6.27}$$

Therefore, the time-averaged profit of normal selfish mining under $\mathsf{DAM}_2^{\mathtt{modified}}$ is equal to:

$$\begin{aligned} &\mathtt{Profit}_{\mathcal{A}}(\pi^{\mathtt{SM}}, \mathsf{DAM}_2^{\mathtt{modified}}) = \\ &\lambda K \Big(\mathtt{RelRev}\big(\pi^{\mathtt{SM}}\big) M_{\mathtt{SM}}^{\mathtt{main\text{-}chain}} \Big) - \alpha_{\mathcal{A}} c_{\mathcal{A}} \ . \end{aligned} \tag{6.28}$$

6.6.4.3 Profitability of Normal Selfish Mining under the Modified DAM

As can be seen in equations 6.25 and 6.28, the time-averaged profit of normal selfish mining under both versions of the modified DAM is the same. In this section, we show that the time-averaged profit of normal selfish mining under the modified DAM is less than the honest strategy time-averaged profit, which can be calculated as $\texttt{Profit}_{\mathcal{A}}(\pi^{\texttt{H}}) = \lambda K \alpha_{\mathcal{A}} - \alpha_{\mathcal{A}} c_{\mathcal{A}}$. To prove this claim, we need to show that:

$$\texttt{RelRev}\big(\pi^{\texttt{SM}}\big) M_{\texttt{SM}}^{\texttt{main-chain}} \leq \alpha_{\mathcal{A}} \ . \tag{6.29}$$

Let $M_{\texttt{SM},\mathcal{A}}^{\texttt{main-chain}}$ and $M_{\texttt{SM},\mathcal{H}}^{\texttt{main-chain}}$ denote the normalized adversarial and honest mining power share extending the main chain during the selfish mining attack, respectively. Therefore, the total power share extending the main chain during the attack is equal to $M_{\texttt{SM}}^{\texttt{main-chain}} = M_{\texttt{SM},\mathcal{A}}^{\texttt{main-chain}} + M_{\texttt{SM},\mathcal{H}}^{\texttt{main-chain}}$. The attacker's relative revenue under selfish mining can be obtained as follows:

$$\texttt{RelRev}\big(\pi^{\texttt{SM}}\big) = \frac{M_{\texttt{SM},\mathcal{A}}^{\texttt{main-chain}}}{M_{\texttt{SM},\mathcal{A}}^{\texttt{main-chain}} + M_{\texttt{SM},\mathcal{H}}^{\texttt{main-chain}}} \ . \tag{6.30}$$

Therefore,

$$\texttt{RelRev}\big(\pi^{\texttt{SM}}\big) M_{\texttt{SM}}^{\texttt{main-chain}} = M_{\texttt{SM},\mathcal{A}}^{\texttt{main-chain}} \ . \tag{6.31}$$

Note that due to the selfish mining attack, some of the adversarial blocks may get orphaned, and as a result $M_{\texttt{SM},\mathcal{A}}^{\texttt{main-chain}} \leq \alpha_{\mathcal{A}}$. This proves the correctness of equation 6.29.

6.7 Smart Intermittent Selfish Mining (Version 2)

In the previous section, we showed that the normal selfish mining attack is unprofitable under the modified DAM. In this section, we present a new version of the selfish mining attack called smart intermittent selfish mining (version 2), which can be profitable even under the modified version of DAM. This attack can be considered as the combination of smart mining [16] and ISM. In the smart intermittent selfish mining (version 2) denoted by SISM2, during $\texttt{epoch}_{\texttt{even}} \in \{\texttt{epoch}_0, \texttt{epoch}_2, \cdots\}$, attacker $\mathcal{A}$ divides his mining power share $\alpha_{\mathcal{A}}$ into two parts: the idle mining power and the honest mining power. We assume the attacker's idle mining power share and honest mining power share are equal to $e\alpha_{\mathcal{A}}$ and $(1-e)\alpha_{\mathcal{A}}$, where $0 \leq e \leq 1$. However, during $\texttt{epoch}_{\texttt{odd}} \in \{\texttt{epoch}_1, \texttt{epoch}_3, \cdots\}$, attacker $\mathcal{A}$ uses all his mining power share to perform selfish mining. Our goal is to show that SISM2 can be more profitable than honest mining under the modified DAM. Note that for the analysis of this section, we assume that all the orphan blocks are reported, and thus, incorporated in the modified DAM.

6.7.1 Analysis under $\mathsf{DAM}_1^{\texttt{modified}}$

The average revenue gained by the attacker in $\texttt{epoch}_{\texttt{even}}$ and his mining cost in $\texttt{epoch}_{\texttt{even}}$ can be obtained as follows:

$$\begin{aligned} R_{\mathcal{A}}(\texttt{epoch}_{\texttt{even}}) &= \frac{(1-e)\alpha_{\mathcal{A}}}{1-e\alpha_{\mathcal{A}}} \cdot LK \ , \\ C_{\mathcal{A}}(\texttt{epoch}_{\texttt{even}}) &= (1-e)\alpha_{\mathcal{A}} c_{\mathcal{A}} \cdot t(\texttt{epoch}_{\texttt{even}}) \ . \end{aligned} \tag{6.32}$$

Note that due to the selfish mining in $\texttt{epoch}_{\texttt{odd}}$ and the definition of epoch in $\mathsf{DAM}_1^{\texttt{modified}}$, the number of main-chain blocks in $\texttt{epoch}_{\texttt{odd}}$ is equal to $M_{\texttt{SM}}^{\texttt{main-chain}} L$. As a result, the total amount of distributed reward in $\texttt{epoch}_{\texttt{odd}}$ is equal to $M_{\texttt{SM}}^{\texttt{main-chain}} LK$. The average revenue gained by the attacker in $\texttt{epoch}_{\texttt{odd}}$ and his mining cost in $\texttt{epoch}_{\texttt{odd}}$ can be obtained as follows.

$$\begin{aligned} R_{\mathcal{A}}(\texttt{epoch}_{\texttt{odd}}) &= \texttt{RelRev}\big(\pi^{\texttt{SM}}\big) \cdot M_{\texttt{SM}}^{\texttt{main-chain}} LK \ , \\ C_{\mathcal{A}}(\texttt{epoch}_{\texttt{odd}}) &= \alpha_{\mathcal{A}} c_{\mathcal{A}} \cdot t(\texttt{epoch}_{\texttt{odd}}) \ . \end{aligned} \tag{6.33}$$

Note that $M_{\texttt{odd}}^{\texttt{total}} = 1$ and $M_{\texttt{even}}^{\texttt{total}} = 1 - e\alpha_{\mathcal{A}}$. According to the design of $\mathsf{DAM}_1^{\texttt{modified}}$, the duration of $\texttt{epoch}_{\texttt{even}}$ and $\texttt{epoch}_{\texttt{odd}}$ can be calculated as follows:

$$t(\texttt{epoch}_{\texttt{even}}) = t^{\texttt{ideal}} \cdot \frac{M_{\texttt{odd}}^{\texttt{total}}}{M_{\texttt{even}}^{\texttt{total}}} = \frac{L}{\lambda(1-e\alpha_{\mathcal{A}})} \ , \tag{6.34}$$

and

$$t(\texttt{epoch}_{\texttt{odd}}) = t^{\texttt{ideal}} \cdot \frac{M_{\texttt{even}}^{\texttt{total}}}{M_{\texttt{odd}}^{\texttt{total}}} = \frac{L(1-e\alpha_{\mathcal{A}})}{\lambda} \ . \tag{6.35}$$

Therefore, the time-averaged profit of SISM2 under $\mathsf{DAM}_1^{\texttt{modified}}$ is equal to:

$$\begin{aligned} &\texttt{Profit}_{\mathcal{A}}(\pi^{\texttt{SISM2}}, \mathsf{DAM}_1^{\texttt{modified}}) = \\ &\frac{\lambda K \Big(\texttt{RelRev}\big(\pi^{\texttt{SM}}\big) M_{\texttt{SM}}^{\texttt{main-chain}} + \frac{(1-e)\alpha_{\mathcal{A}}}{1-e\alpha_{\mathcal{A}}} \Big)}{1 - e\alpha_{\mathcal{A}} + \frac{1}{1-e\alpha_{\mathcal{A}}}} \\ &+ \frac{e\alpha_{\mathcal{A}} c_{\mathcal{A}} \cdot \frac{1}{1-e\alpha_{\mathcal{A}}}}{1 - e\alpha_{\mathcal{A}} + \frac{1}{1-e\alpha_{\mathcal{A}}}} - \alpha_{\mathcal{A}} c_{\mathcal{A}} \ . \end{aligned} \tag{6.36}$$

6.7.2 Analysis under $\mathsf{DAM}_2^{\texttt{modified}}$

The average revenue gained by the attacker in $\texttt{epoch}_{\texttt{even}}$ and his mining cost in $\texttt{epoch}_{\texttt{even}}$ can be obtained as follows:

$$\begin{aligned} R_{\mathcal{A}}(\texttt{epoch}_{\texttt{even}}) &= \frac{(1-e)\alpha_{\mathcal{A}}}{1-e\alpha_{\mathcal{A}}} \cdot LK \ , \\ C_{\mathcal{A}}(\texttt{epoch}_{\texttt{even}}) &= (1-e)\alpha_{\mathcal{A}} c_{\mathcal{A}} \cdot t(\texttt{epoch}_{\texttt{even}}) \ . \end{aligned} \tag{6.37}$$

According to the definition of epoch in $\mathsf{DAM}_2^{\texttt{modified}}$, the number of main-chain blocks and the total amount of distributed reward in $\texttt{epoch}_{\texttt{odd}}$ are equal to L and LK, respectively. The average revenue gained by the attacker in $\texttt{epoch}_{\texttt{odd}}$ and his mining cost in $\texttt{epoch}_{\texttt{odd}}$ can be obtained as follows.

$$\begin{aligned} R_{\mathcal{A}}(\texttt{epoch}_{\texttt{odd}}) &= \texttt{RelRev}\big(\pi^{\texttt{SM}}\big) \cdot LK \ , \\ C_{\mathcal{A}}(\texttt{epoch}_{\texttt{odd}}) &= \alpha_{\mathcal{A}} c_{\mathcal{A}} \cdot t(\texttt{epoch}_{\texttt{odd}}) \ . \end{aligned} \tag{6.38}$$

Note that $M_{\texttt{odd}}^{\texttt{total}} = 1$, $M_{\texttt{odd}}^{\texttt{main-chain}} = M_{\texttt{SM}}^{\texttt{main-chain}}$, and $M_{\texttt{even}}^{\texttt{total}} = M_{\texttt{even}}^{\texttt{main-chain}} = 1 - e\alpha_{\mathcal{A}}$. According to the design of $\mathsf{DAM}_2^{\texttt{modified}}$, the duration of $\texttt{epoch}_{\texttt{even}}$ and $\texttt{epoch}_{\texttt{odd}}$ can be calculated as follows:

$$t(\texttt{epoch}_{\texttt{even}}) = t^{\texttt{ideal}} \cdot \frac{M_{\texttt{odd}}^{\texttt{total}}}{M_{\texttt{even}}^{\texttt{main-chain}}} = \frac{L}{\lambda(1-e\alpha_{\mathcal{A}})} \ , \tag{6.39}$$

and

$$t(\texttt{epoch}_{\texttt{odd}}) = t^{\texttt{ideal}} \cdot \frac{M_{\texttt{even}}^{\texttt{total}}}{M_{\texttt{odd}}^{\texttt{main-chain}}} = \frac{L(1-e\alpha_{\mathcal{A}})}{\lambda M_{\texttt{SM}}^{\texttt{main-chain}}} \ . \tag{6.40}$$

Therefore, the time-averaged profit of SISM2 under $\mathsf{DAM}_2^{\texttt{modified}}$ is equal to:

$$\begin{aligned} &\texttt{Profit}_{\mathcal{A}}(\pi^{\texttt{SISM2}}, \mathsf{DAM}_2^{\texttt{modified}}) = \\ &\frac{\lambda K\Big(\texttt{RelRev}\big(\pi^{\texttt{SM}}\big) + \frac{(1-e)\alpha_{\mathcal{A}}}{1-e\alpha_{\mathcal{A}}}\Big) + e\alpha_{\mathcal{A}} c_{\mathcal{A}} \cdot \frac{1}{1-e\alpha_{\mathcal{A}}}}{\frac{1-e\alpha_{\mathcal{A}}}{M_{\texttt{SM}}^{\texttt{main-chain}}} + \frac{1}{1-e\alpha_{\mathcal{A}}}} \\ &- \alpha_{\mathcal{A}} c_{\mathcal{A}} \ . \end{aligned} \tag{6.41}$$

6.7.3 Profitability of SISM2 under the Modified DAM

In this section, we show that SISM2 can be more profitable than honest mining even under the modified DAM (when the orphan blocks are reported). We first

define the profitability advantage of strategy π, which is denoted by $\mathtt{P}^{\mathtt{adv}}(\pi)$, as follows:

$$\mathtt{P}^{\mathtt{adv}}(\pi) = \frac{\mathtt{Profit}_{\mathcal{A}}(\pi) - \mathtt{Profit}_{\mathcal{A}}(\pi^{\mathtt{H}})}{\lambda K} \quad . \tag{6.42}$$

$\mathtt{P}^{\mathtt{adv}}(\pi) > 0$ indicates that strategy π is more profitable than honest mining. Depending on the profitability factor $\omega_{\mathcal{A}}$, the mining power share $\alpha_{\mathcal{A}}$, the communication capability $\gamma_{\mathcal{A}}$, and the version of the modified DAM, the amount of attacker's idle mining power share in $\mathtt{epoch}_{\mathtt{even}}$ that maximizes the SISM2 time-averaged profit may vary. The maximum amount of profitability advantage of SISM2 under $\mathsf{DAM}_1^{\mathtt{modified}}$ and under $\mathsf{DAM}_2^{\mathtt{modified}}$ are depicted in Figure 6.4 and Figure 6.5, respectively. These figures represent the profitability advantage as a function of $\alpha_{\mathcal{A}}$ and $\omega_{\mathcal{A}}$ for two distinct values of communication capability. Note that to draw these figures, we have used the optimal selfish mining strategy [35] to calculate the attacker's relative revenue and main-chain effective power in $\mathtt{epoch}_{\mathtt{odd}}$. As can be seen, at some points in the maps depicted in Figure 6.4 and Figure 6.5, $\mathtt{P}^{\mathtt{adv}}(\pi^{\mathtt{SISM2}}) > 0$, showing that selfish mining can be profitable even when the orphan blocks are incorporated into the DAM.

Intuition behind SISM2 profitability: SISM2 profitability lies in the fact that the idle mining power in $\mathtt{epoch}_{\mathtt{even}}$ results in a decrease in the mining difficulty of $\mathtt{epoch}_{\mathtt{odd}}$. As a result, the attacker can mine a new block in a shorter period of time during $\mathtt{epoch}_{\mathtt{odd}}$, which leads to collecting a greater reward. If the extra collected reward in $\mathtt{epoch}_{\mathtt{odd}}$ can compensate for the loss of being idle in $\mathtt{epoch}_{\mathtt{even}}$, the attack becomes profitable. For the attackers whose profitability factor $\omega_{\mathcal{A}}$ is relatively low, being idle does not cause a huge profit loss, and consequently, SISM2 can be more profitable than honest mining. At the time of writing on February 26, 2024, the average Bitcoin profitability factor is equal to 1.071 [1]. In Appendix C of [37], we show that under $\mathsf{DAM}_1^{\mathtt{modified}}$, SISM2 cannot be more profitable than smart honest mining [16], in which the miner

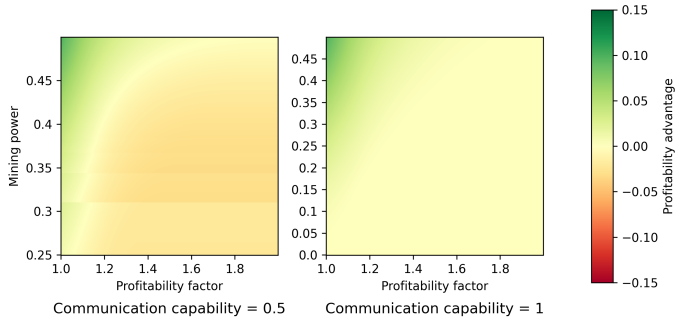


Figure 6.4: Profitability advantage of SISM2 under $\mathsf{DAM}_1^{\mathtt{modified}}$

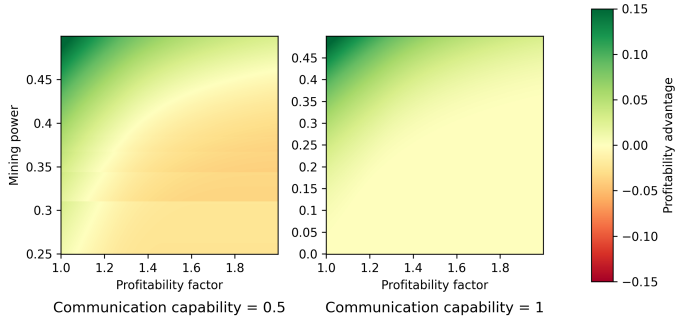


Figure 6.5: Profitability advantage of SISM2 under $\mathsf{DAM}_2^{\texttt{modified}}$

switches between honest mining and being idle. However, under $\mathsf{DAM}_2^{\texttt{modified}}$, SISM2 can even be more profitable than smart honest mining.

6.8 Orphan Exclusion Attack

As already shown in the previous section, an intelligently executed selfish mining attack can be more profitable than honest mining even if orphan blocks are considered in the DAM. In the analysis of the previous section, we have assumed that all the orphan blocks get reported by the honest miners and subsequently incorporated in the modified DAM. However, the attacker can impose an attack, we refer to as the orphan exclusion attack (OEA), that stops honest miners from reporting some of the orphan blocks. By performing the orphan exclusion attack, a selfish miner can increase his time-averaged profit under the modified DAM. In this section, we explain the orphan exclusion attack and try to analyze for how long the attacker can prevent the honest miners from reporting the orphan blocks.

6.8.1 Attack Description

In this attack, the attacker tries to orphan a set of consecutive honest blocks at the end of each epoch including the last honest block of the epoch. Whenever a few blocks are left to the end of each epoch, the attacker starts orphaning the public chain. To do so, the attacker separates his private chain from the public chain, i.e., forks the public chain, and tries to extend his private chain. The attack is considered to be successful if the following two conditions are satisfied:

1. Starting from the fork point, the attacker's private chain manages to orphan the public chain.
2. The last block of the epoch is included in the attacker's private chain.

By performing this attack, the attacker manages to orphan some of the honest blocks that will never be reported inside the other honest blocks of the main chain. The scenario of the orphan exclusion attack is depicted in Figure 6.6. Assume the attacker starts the attack when L_1^{OEA} blocks are left to the end of epoch i and manages to orphan the public chain after L^{OEA} blocks, where $L_1^{\mathsf{OEA}} \leq L^{\mathsf{OEA}}$. We use L^{OEA} to denote the length of the orphan exclusion attack. As a result of the attack, L_1^{OEA} blocks get orphaned at the end of epoch i. Because there is no honest block included in the main chain from the start of the attack till the end of epoch i, these L_1^{OEA} orphaned honest blocks cannot be reported and consequently cannot be incorporated in the DAM deciding the difficulty of epoch $i+1$. In addition, if assuming $L^{\mathsf{OEA}} = L_1^{\mathsf{OEA}} + L_2^{\mathsf{OEA}}$, the attacker manages to orphan L_2^{OEA} honest blocks at the start of epoch $i+1$. According to condition (3) in the uncle definition, prior to reporting the L_2^{OEA} honest blocks orphaned at the start of epoch $i+1$, the honest miners should report their ancestors, i.e., the L_1^{OEA} honest blocks orphaned at the end of epoch i. However, according to condition (1) in the uncle definition, the honest blocks in epoch $i+1$ cannot report the orphaned blocks of the previous epochs. As a result, these L_2^{OEA} orphaned honest blocks cannot be reported by the honest blocks in epoch $i+1$ and consequently cannot be incorporated in the DAM deciding the difficulty of epoch $i+2$. Therefore, by imposing a successful orphan exclusion attack, in total, L^{OEA} orphaned honest blocks cannot be reported in the current and next epochs. The orphan exclusion reduces the mining difficulty calculated by the modified DAM in favor of the attacker, which can result in an increase in the selfish mining time-averaged profit.

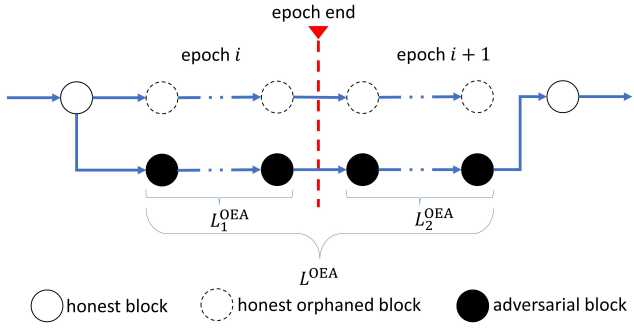


Figure 6.6: Orphan exclusion attack

The attacker should decide on the starting and ending time of the orphan exclusion attack. The greater the length of the orphan exclusion attack, the more profitable the selfish mining. However, increasing the length of the orphan exclusion attack reduces the success probability of the attack. Assume the attacker starts the attack when a few blocks are left to the end of an epoch. If during the attack until the end of the epoch, the attacker's private chain always has a lead over the public chain, the attacker can easily orphan the public chain and finish the attack successfully. However, there is a possibility that in the middle of the attack and before reaching the end of the epoch, the public chain gets a lead over the attacker's private chain. In this situation, the attacker should decide whether he wants to continue mining on top of his private chain or stop the orphan exclusion attack and join the public chain. On the one hand, if the attacker decides to stop the attack before reaching the end of the epoch, regardless of whether he managed to orphan the honest blocks or not, the attack cannot cause a reduction in the difficulty level specified by the upcoming modified DAM because the remaining honest blocks added to the main chain before the end of the epoch can report the orphaned blocks. On the other hand, if the attacker decides to continue mining on top of his private chain, he will risk losing more blocks. Therefore, the attacker should devise a strategy for the orphan exclusion attack that can maximize the length of the orphan exclusion attack.

We aim to calculate the attacker's time-averaged profit under the modified DAMs while performing both selfish mining and orphan exclusion attacks. The first step towards calculating the attacker's time-averaged profit is to calculate the length of the orphan exclusion attack. This length represents the period during which the attacker can prevent the honest miners from adding an honest block to the blockchain and reporting the orphaned blocks. It is obvious that if the attacker's mining power is less than the honest miners', the attacker cannot continue orphaning all the honest blocks forever, and there will be an honest block that gets added to the blockchain and terminates the orphan exclusion attack [8]. Note that the concept of honest block exclusion is similar to the suppression concept introduced in [21], where the attacker tries to suppress the honest blocks and put them out of the main chain. While the authors in [21] have focused on calculating the number of suppressed honest blocks, in this paper, we try to find the length of **consecutive** suppressed honest blocks.

6.8.2 The Length of the Orphan Exclusion Attack

In this section, we aim to calculate the average length of the orphan exclusion attack performed by attacker $\mathcal{A}$ at the end of each epoch under the modified DAMs. To calculate the length of the longest possible orphan exclusion attack,

we assume attacker $\mathcal{A}$ enjoys the highest communication capability and can predict future block miners. We first explain the impact of communication capability and predictability on the orphan exclusion attack.

Definition 6.8.1 (Mining sequence)**.** A mining sequence, which is denoted by S, is an ordered list of blocks that specifies the miner of each block. B_i^{H} ($B_i^{\mathcal{A}}$) is used to represent the ith honest (adversarial) block in the mining sequence S. The blocks in S are ordered by the time at which they are mined.

Note that not all the blocks of S are included in the main chain since some of them may get orphaned. We assume all the orphans in S are caused by the attack, namely, there is no naturally orphaned block in S.

Predictive capability: The attacker can predict the elements of mining sequence S in advance. In other words, the attacker knows which of the upcoming blocks are mined either by himself or by the honest miners.

Note that, in Bitcoin, no miner knows who is the next block proposer until they solve the puzzle or receive a new block from the blockchain network, i.e., no miners enjoy the predictive capability. In Appendix D of [37], we discuss how an attacker without the predictive capability can perform the OEA attack. In this section, to calculate the maximum length of the orphan exclusion attack, we assume that $\mathcal{A}$ enjoys the predictive capability and his communication capability ($\gamma_{\mathcal{A}}$) is equal to 1. In the following, we argue that in such a scenario, $\mathcal{A}$ can impose the longest possible orphan exclusion attack at the end of each epoch: Possessing the predictive capability helps $\mathcal{A}$ not only to successfully finish all the orphan exclusion attacks but to impose the longest possible attack. Consider the mining sequence $S = \{B_1^{\mathrm{H}}, B_1^{\mathcal{A}}, B_2^{\mathrm{H}}, B_3^{\mathrm{H}}, B_2^{\mathcal{A}}, B_3^{\mathcal{A}}, B_4^{\mathcal{A}}, B_4^{\mathrm{H}}, B_5^{\mathrm{H}}, \cdots\}$ depicted in Figure 6.7. Assume a real-world attacker, who has mined block $B_1^{\mathcal{A}}$, decides

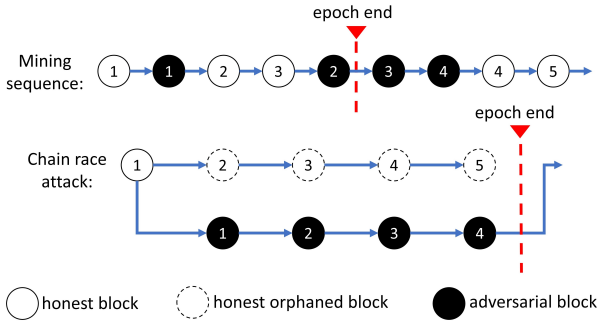


Figure 6.7: Comparison of the orphan exclusion attack imposed by a real-world attacker and $\mathcal{A}$

Table 6.1: The length of the orphan exclusion attack

mining power share	0.25	0.3	0.35	0.4	0.45
$L^{\mathsf{OEA}}(\mathsf{DAM}_1^{\mathtt{modified}})$	1.45	2.99	6.83	18.36	87.02
$L^{\mathsf{OEA}}(\mathsf{DAM}_2^{\mathtt{modified}})$	15.48	23.87	40.02	78.67	225.79

to keep the block secret and start the orphan exclusion attack. Since the next block, i.e., B_2^{H}, is mined by the honest miners, the block race situation occurs. In this case, the real-world attacker should decide whether he wants to publish block $B_1^{\mathcal{A}}$ or continue the orphan exclusion attack. Continuing the attack can increase the risk of losing mined blocks for the real-world attacker. However, since $\mathcal{A}$ is aware of the mining sequence, he knows he will eventually win the chain race and successfully finish the attack. Therefore, at the end of each epoch, $\mathcal{A}$ either does not start the orphan exclusion attack or imposes a successful one. Moreover, at the end of each epoch, there is a possibility that an attacker can impose successful orphan exclusion attacks of different lengths. For instance, consider the mining sequence depicted in Figure 6.7. The attacker can start a successful orphan exclusion attack both at block $B_1^{\mathcal{A}}$ and block $B_2^{\mathcal{A}}$. However, the length of the former attack is equal to 4 and the length of the latter one is equal to 3 (assuming a long set of consecutive honest blocks after block B_5^{H}). This shows that $\mathcal{A}$ can specify the start and end points of the attack in a way that maximizes the length of the attack. Having $\gamma_{\mathcal{A}} = 1$ helps $\mathcal{A}$ to orphan the maximum possible number of honest blocks (the same number as the orphan exclusion attack length).

A comprehensive analysis for calculating the length of the orphan exclusion attack under $\mathsf{DAM}_1^{\mathtt{modified}}$ and $\mathsf{DAM}_2^{\mathtt{modified}}$ are presented in Appendix E and Appendix F of [37], respectively. Table 6.1 presents the average length of the orphan exclusion attack under $\mathsf{DAM}_1^{\mathtt{modified}}$ and $\mathsf{DAM}_2^{\mathtt{modified}}$ performed by attacker $\mathcal{A}$, who enjoys the predictive capability and the highest possible communication capability. The results presented in Table 6.1 show that the length of orphan exclusion attack under $\mathsf{DAM}_2^{\mathtt{modified}}$ is longer than that under $\mathsf{DAM}_1^{\mathtt{modified}}$. This indicates that selfish mining accompanied by the orphan exclusion attack can be more profitable under $\mathsf{DAM}_2^{\mathtt{modified}}$ compared to that under $\mathsf{DAM}_1^{\mathtt{modified}}$.

6.9 Selfish Mining Accompanied by the Orphan Exclusion Attack

In this section, we aim to assess the effect of the orphan exclusion attack on selfish mining profitability under the modified DAM. As shown in Section 6.6.4,

the normal selfish mining attack in which the attacker follows selfish mining continuously for all the epochs is not profitable under the modified DAM. Here, we show that applying the orphan exclusion attack can make the normal selfish mining attack profitable under the modified DAM. For our analysis in this section, we assume that the attacker manages to perform a successful orphan exclusion attack at the end of each epoch, where the attack length is denoted by L^{OEA}. This indicates that the last L^{OEA} blocks of the main chain in each epoch are adversarial, and L^{OEA} honest blocks get orphaned without being reported in each epoch.

6.9.1 Analysis under $\mathsf{DAM}_1^{\mathtt{modified}}$

We first calculate the average revenue gained by the attacker in each epoch and his mining cost. We denote by L_{epoch} the total number of both orphaned and main-chain blocks mined during one epoch under $\mathsf{DAM}_1^{\mathtt{modified}}$. Note that out of the first $L_{\text{epoch}} - L^{\mathsf{OEA}}$ blocks of the epoch, only $M_{\mathtt{SM}}^{\mathtt{main\text{-}chain}}(L_{\text{epoch}} - L^{\mathsf{OEA}})$ blocks are added to the main chain, out of which $\mathtt{RelRev}\big(\pi^{\mathtt{SM}}\big) M_{\mathtt{SM}}^{\mathtt{main\text{-}chain}}(L_{\text{epoch}} - L^{\mathsf{OEA}})$ blocks are adversarial. The last L^{OEA} blocks of the epoch that are added to the main chain are adversarial blocks.

$$
\begin{aligned}
&R_{\mathcal{A}}(\mathtt{epoch}_i) = \\
&\mathtt{RelRev}\big(\pi^{\mathtt{SM}}\big) M_{\mathtt{SM}}^{\mathtt{main\text{-}chain}}(L_{\text{epoch}} - L^{\mathsf{OEA}})K + L^{\mathsf{OEA}}K \ , \\
&C_{\mathcal{A}}(\mathtt{epoch}_i) = \alpha_{\mathcal{A}} c_{\mathcal{A}} \cdot t(\mathtt{epoch}_i) \ .
\end{aligned} \tag{6.43}
$$

According to the design of the modified DAM introduced in section 6.6.2, the target of $\mathtt{epoch}_i$ can be calculated as follows:

$$
\mathsf{TGT}_i = \mathsf{TGT}_{i-1} \frac{t(\mathtt{epoch}_{i-1})}{t^{\mathtt{ideal}}} \ . \tag{6.44}
$$

Since the total amount of mining power and the miners' strategy are consistent throughout the whole epochs, the duration and the mining target of all the epochs would be the same, i.e., $\mathsf{TGT}_i = \mathsf{TGT}_{i-1}$ and $t(\mathtt{epoch}_i) = t(\mathtt{epoch}_{i-1})$. Therefore, by using equation 6.44, the duration of each epoch can be calculated as follows:

$$
t(\mathtt{epoch}_i) = t^{\mathtt{ideal}} = \frac{L_{\text{epoch}}}{\lambda} \ . \tag{6.45}
$$

Therefore, the time-averaged profit of normal selfish mining accompanied by the orphan exclusion attack under $\mathsf{DAM}_1^{\texttt{modified}}$ is equal to:

$$\begin{aligned}&\texttt{Profit}_{\mathcal{A}}(\pi^{\texttt{SM-OEA}}, \mathsf{DAM}_1^{\texttt{modified}}) = \\ &\lambda K\Big(\texttt{RelRev}\big(\pi^{\texttt{SM}}\big) M_{\texttt{SM}}^{\texttt{main-chain}}\big(\frac{L_{\text{epoch}} - L^{\mathsf{OEA}}}{L_{\text{epoch}}}\big) + \frac{L^{\mathsf{OEA}}}{L_{\text{epoch}}}\Big) \\ &- \alpha_{\mathcal{A}} c_{\mathcal{A}} \ .\end{aligned} \tag{6.46}$$

6.9.2 Analysis under $\mathsf{DAM}_2^{\texttt{modified}}$

We first calculate the average revenue gained by the attacker in each epoch and his mining cost. We denote by L_{epoch} the total number of main-chain blocks mined during one epoch under $\mathsf{DAM}_2^{\texttt{modified}}$. Note that out of the first $L_{\text{epoch}} - L^{\mathsf{OEA}}$ main-chain blocks, only $\texttt{RelRev}\big(\pi^{\texttt{SM}}\big)(L_{\text{epoch}} - L^{\mathsf{OEA}})$ blocks are adversarial. The last L^{OEA} blocks of the main chain are adversarial blocks.

$$\begin{aligned}&R_{\mathcal{A}}(\texttt{epoch}_i) = \texttt{RelRev}\big(\pi^{\texttt{SM}}\big)(L_{\text{epoch}} - L^{\mathsf{OEA}})K + L^{\mathsf{OEA}}K \ , \\ &C_{\mathcal{A}}(\texttt{epoch}_i) = \alpha_{\mathcal{A}} c_{\mathcal{A}} \cdot t(\texttt{epoch}_i) \ .\end{aligned} \tag{6.47}$$

According to the design of the modified DAM introduced in section 6.6.3, the target of $\texttt{epoch}_i$ can be calculated as follows:

$$\mathsf{TGT}_i = \mathsf{TGT}_{i-1} \frac{t(\texttt{epoch}_{i-1})}{\frac{\mathsf{CNT}_i^{\texttt{orphan-total}} + L_{\text{epoch}} - L^{\mathsf{OEA}}}{L_{\text{epoch}}} t^{\texttt{ideal}}} \ , \tag{6.48}$$

where $\mathsf{CNT}_i^{\texttt{orphan-total}}$ represents the total number of both reported and non-reported orphan blocks in $\texttt{epoch}_i$. Since the total amount of mining power and the miners' strategy are consistent throughout the whole epochs, the duration and the mining target of all the epochs would be the same, i.e., $\mathsf{TGT}_i = \mathsf{TGT}_{i-1}$ and $t(\texttt{epoch}_i) = t(\texttt{epoch}_{i-1})$. Therefore, by using equation 6.48, the duration of each epoch can be calculated as follows:

$$t(\texttt{epoch}_i) = \frac{\mathsf{CNT}_i^{\texttt{orphan-total}} + L_{\text{epoch}} - L^{\mathsf{OEA}}}{L_{\text{epoch}}} t^{\texttt{ideal}} \ . \tag{6.49}$$

As $M_{\texttt{SM}}^{\texttt{main-chain}} = \frac{L_{\text{epoch}}}{\mathsf{CNT}_i^{\texttt{orphan-total}} + L_{\text{epoch}}}$, we have:

$$t(\texttt{epoch}_i) = \frac{L_{\text{epoch}} - M_{\texttt{SM}}^{\texttt{main-chain}} L^{\mathsf{OEA}}}{M_{\texttt{SM}}^{\texttt{main-chain}} L_{\text{epoch}}} \cdot \frac{L_{\text{epoch}}}{\lambda} \ . \tag{6.50}$$

Therefore, the time-averaged profit of normal selfish mining accompanied by the orphan exclusion attack under $\mathsf{DAM}_2^{\texttt{modified}}$ is equal to:

$$\begin{aligned}&\texttt{Profit}_{\mathcal{A}}(\pi^{\texttt{SM-OEA}}, \mathsf{DAM}_2^{\texttt{modified}}) = \\ &\lambda K\Big(\frac{M_{\texttt{SM}}^{\texttt{main-chain}} L_{\text{epoch}}}{L_{\text{epoch}} - M_{\texttt{SM}}^{\texttt{main-chain}} L^{\mathsf{OEA}}}\Big) \times \\ &\Big(\texttt{RelRev}\big(\pi^{\texttt{SM}}\big)\big(\frac{L_{\text{epoch}} - L^{\mathsf{OEA}}}{L_{\text{epoch}}}\big) + \frac{L^{\mathsf{OEA}}}{L_{\text{epoch}}}\Big) - \alpha_{\mathcal{A}} c_{\mathcal{A}} \ .\end{aligned} \tag{6.51}$$

6.9.3 Profitability of Normal Selfish Mining Accompanied by the Orphan Exclusion Attack under the Modified DAM

Due to the orphan exclusion attack, the mining difficulty of the subsequent epoch decreases, which results in an increase in selfish mining profitability. In Figure 6.8, the profitability advantage of normal selfish mining accompanied by the orphan exclusion attack is depicted as a function of epoch length L_{epoch} for the attacker with $\alpha_{\mathcal{A}} = 0.4$ and $\gamma_{\mathcal{A}} = 1$ under both versions of the modified DAM. As can be seen in Figure 6.8, $\mathsf{P}^{\texttt{adv}}(\pi^{\texttt{SM-OEA}}) > 0$, indicating that the normal selfish mining attack can be profitable even if the orphans are incorporated in the DAM. Additionally, it can be observed that a reduction in the epoch length can result in a more destructive selfish mining attack. This highlights a trade-off in the design of the DAM between resistance to selfish mining and sensitivity to hash rate fluctuations.

6.10 Discussion

There has been significant debate about the lack of occurrence of selfish mining attacks against Bitcoin. Although selfish mining is profitable under the current DAM, Bitcoin has operated smoothly for over a decade without strong evidence of such attacks. This raises the question of why miners avoid selfish mining despite its potential profitability. Beyond the high mining power required for profitable selfish mining, another argument is miners' goodwill to secure the network [5] and preserve Bitcoin's reputation. While a miner could earn more through selfish mining, detection of such an attack could harm Bitcoin's reputation and decrease its value. However, if an attacker maintains an acceptable forking rate, the attack may go undetected [24]. Additionally, as protocols evolve, the cryptocurrency may recover its price, making past selfish mining rewards valuable again. Moreover, rising energy prices or falling Bitcoin values may lead mining pools to find their earnings insufficient to cover

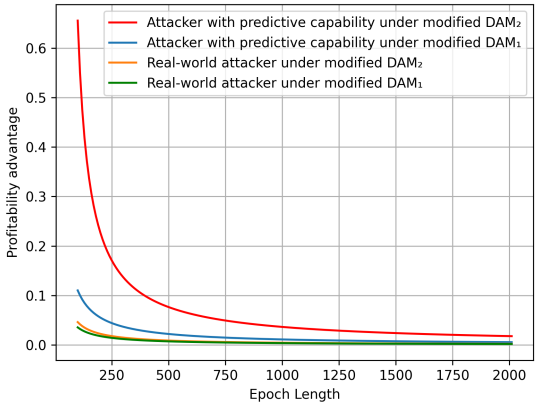


Figure 6.8: The profitability advantage of $\pi^{\texttt{SM-OEA}}$ under the modified DAM

costs, potentially forcing them to stop mining or abandon the honest strategy. In such cases, miners might prioritize profits over the coin's reputation and consider deviating from the protocol. These possibilities indicate that the lack of selfish mining attacks in Bitcoin thus far does not prevent future occurrences, emphasizing the need for thorough research into potential countermeasures.

6.10.1 Selfish Mining Countermeasures

The countermeasures proposed in the literature can be divided into two general categories: those that aim to improve chain quality (by reducing the adversarial block ratio), referred to as better-chain-quality protocols, and those that seek to modify the reward distribution or difficulty adjustment mechanisms, known as mechanism-altering protocols [42].

6.10.1.1 Better-Chain-Quality Protocols

In these protocols, the goal is to help miners distinguish between honest and adversarial blocks. A key factor for this distinction is the delay between block generation and its publication on the network. If a block is published immediately after it is generated, it can be considered honest; otherwise, it is viewed as adversarial. However, since the concept of timeliness in blockchains is entirely subjective to each miner's local view of the chain, any selfish mining mitigation relying on time-based metrics—such as block timestamps, transaction times, propagation delays, or block receipt times—faces inherent limitations in

fully addressing the selfish mining threat. While these protocols may work under synchrony, in practice, and under partial asynchrony, it becomes impossible to distinguish whether a block's delay is due to adversarial behavior or temporary network partitioning.

In the freshness-preferred scheme [19], the countermeasure relies on unforgeable timestamps, which necessitates trust in a third party. In schemes [23, 34], the countermeasure is based on information embedded in transactions, such as transaction creation time and expected confirmation height. A sufficient number of adversarial transactions can bypass these countermeasures. Countermeasures that rely on dummy blocks [38], block interval times [32], and in-time blocks [41] to detect immediate block publication face significant challenges under partial asynchrony, where a block may be considered honest by some miners but adversarial by others, depending on their network partition. Under partial asynchrony, some of these approaches may not only fail to mitigate selfish mining but could also threaten the validity of honest blocks that experience propagation delays.

6.10.1.2 Mechanism-Altering Protocols

Mechanism-altering protocols modify how rewards are distributed or how difficulty is adjusted to make selfish mining unprofitable or, at the very least, to reduce its profitability.

The Fruitchain protocol [31] aims to mitigate selfish mining through its reward distribution method. In Fruitchain, a fruit mined on a recent block, a block that is no more than $l_{\texttt{recent}}$ blocks deep, receives a reward. Assuming the maximum length of an adversarial fork during selfish mining is less than $l_{\texttt{recent}}$, these fruits cannot be orphaned and will eventually be included in an honest block in the main chain. Fruitchain mitigates selfish mining to an acceptable extent, but at the cost of longer confirmation times and higher storage costs due to repetitive transactions in fruits. Besides, as shown in [42], Fruitchain performs worse than the current Bitcoin protocol when the attacker's communication capability is zero. This is because, in Fruitchain, blocks have no rewards, allowing even a weak attacker to withhold its blocks in hopes of creating a fork longer than $l_{\texttt{recent}}$, a strategy that is not profitable in the current Bitcoin protocol.

Another important consideration is that Fruitchain can mitigate selfish mining only if the rewards for all fruits are assumed to be the same. In practice, however, to incentivize miners to include transactions in their fruits, the reward for a fruit should depend on the fees of the transactions included in it. Since fruits in Fruitchain protocols are mined in parallel (they can be mined on the same block), it is possible for different fruits to share many transactions in common. This raises the question of which miner should receive the fee for a

transaction included in multiple fruits. One naive design is to divide the fee of a transaction among all the fruit miners who include that transaction; however, this could incentivize miners to include repetitive transactions in their fruits, posing a significant liveness threat to the chain. A more reasonable design would be to award the fee to the miner of the first fruit that includes the transaction. However, since the order of fruits in the Fruitchain protocol is determined by the blocks, any attack that results in reordering fruits can become a profitable strategy. Consequently, a miner is incentivized to orphan blocks by conducting selfish mining to reorder transactions, allowing it to gain higher transaction fees than its fair share.

Another approach to making selfish mining unprofitable is to modify the difficulty adjustment mechanism. Here, we discuss the design of three proposals from the literature: (i) increasing the mining difficulty after the detection of a successful selfish mining attack [43], (ii) adjusting the difficulty of the next epoch as the average of the current epoch's difficulty and the estimated difficulty [4], and (iii) incorporating the count of orphan blocks into the difficulty adjustment [17]. The last approach is the one we analyzed in this paper, and we will further discuss its strengths and limitations in Section 6.10.2.

In the first approach [43], assuming the adjusted mining difficulty under selfish mining is normalized to 1, the authors propose a mitigation strategy to increase the difficulty to a predefined value β, where $1 < \beta < 2$. The intuition behind this approach is to compensate for the reduction in mining difficulty resulting from selfish mining by increasing the difficulty. Although the authors have obtained the optimal value of β for different adversarial mining power shares, the main practical challenge facing this scheme is how miners should estimate the adversarial mining share and subsequently determine the appropriate value for the difficulty-increasing factor β in practice—an issue that remains unanswered. The advantage of reporting orphans is that it provides a metric for miners to estimate the total active mining share in the network, allowing them to adjust the mining difficulty proportionally. This eliminates the need for a predefined value of β that requires consensus among miners. Another point worth mentioning is that the authors assume the difficulty after selfish mining as the normalized difficulty, and based on that, they suggest the mining difficulty should be increased. However, if we normalize the mining difficulty before the attack to one, we only need to keep the mining difficulty fixed and prevent its reduction to make selfish mining unprofitable.

In the second scheme [4], assuming the difficulty of the initial selfish mining epoch is denoted by D_1, and the calculated difficulty based on Bitcoin's current DAM for the subsequent epoch is E_2, the authors propose setting the difficulty for the next epoch as $D_2 = \frac{D_1+E_2}{2}$. The intuition behind this approach is that modifying the DAM in this way slows down the reduction in mining difficulty

resulting from selfish mining. This creates a much longer profit lag, i.e., an initial loss period, which demotivates miners from engaging in selfish mining. The first limitation of this approach is that despite the extended profit lag, the difficulty under long-range selfish mining will eventually decrease to the same level as it would under the current DAM, making selfish mining profitable in the long run. The primary limitation of the proposed DAM is that it cannot distinguish whether the reduced block generation rate is due to an ongoing selfish mining attack or some miners going offline. In the latter case, the proposed DAM would prevent the reduction of mining difficulty, causing the difficulty to remain improperly adjusted based on the active online mining power. Reporting orphaned blocks can help distinguish between a selfish mining attack and miners going offline.

6.10.2 Reporting Orphan Blocks: Can It Solve All the Problems?

In this paper, we introduced two versions of a modified DAM that incorporate the count of orphan blocks when estimating the active mining power share. We demonstrate that these modified DAMs, which can be viewed as the most natural DAMs that include the count of orphan blocks, are not entirely attack-resistant. Attacks such as smart intermittent selfish mining and orphan exclusion attacks can still threaten Bitcoin's security even under these modified DAMs. Despite the existence of such attacks, as shown in Figure 6.8, the modified versions of the DAM introduced in this paper can significantly limit the increase in a selfish miner's time-averaged profit in the real world, particularly when the epoch length is relatively long. This suggests that reporting orphan blocks can discourage payoff-maximizing selfish miners to an acceptable extent, making these modified DAMs a viable countermeasure against selfish mining.

Implementing a solution to defend against selfish mining can be a reasonable decision provided that the overhead caused by the proposed solution does not exceed its benefits. Compared to the Bitcoin DAM, the modified DAM imposes higher communication and storage costs to the Bitcoin network because referring to orphans in the honest blocks would increase the block size. However, firstly since only the hash of orphan blocks is reported, and secondly, due to the very low forking rate in the normal situation, the added network and storage costs by the modified DAM would be negligible. Therefore, by applying the modified DAM, at the cost of a very small increase in communication and storage costs, we can significantly decrease selfish mining profitability. As a comparison between two versions of the modified DAM introduced in this paper, it is worth mentioning that under $\mathsf{DAM}_1^{\texttt{modified}}$, the profitability of the SISM2 attack cannot surpass smart honest mining profitability. However, under

$\mathsf{DAM}_2^{\texttt{modified}}$, SISM2 profitability can surpass smart honest mining profitability. Moreover, the average length of the orphan exclusion attack under $\mathsf{DAM}_1^{\texttt{modified}}$ is less than that under $\mathsf{DAM}_2^{\texttt{modified}}$. Therefore, between these two versions, $\mathsf{DAM}_1^{\texttt{modified}}$ seems to be the superior choice to implement.

Although the modified DAM can significantly reduce the profitability of selfish mining, it has its limitations and cannot counter all adversarial mining strategies. A Byzantine adversary, who is unconcerned about profitability, can execute selfish mining only to harm honest miners. While selfish mining may not yield profits under the modified DAM, it can negatively impact honest miners' profitability by lowering the block generation rate and reducing chain quality. This raises the concern that, in practice, rational non-adversarial miners may avoid reporting orphans to minimize the attack's impact. Additionally, other mining strategies, such as smart mining [16] and coin hopping, remain profitable under the modified DAM. In coin hopping, the miner shifts between mining on different blockchains that use the same PoW mechanism, such as Bitcoin and Bitcoin Cash. These strategies can manipulate mining difficulty to increase profitability without generating orphan blocks. As a result, in the absence of evidence indicating an attack, the modified DAM is unable to detect and mitigate these strategies. For a detailed analysis of the coin-hopping attack profitability, readers are referred to [18, 20, 22].

6.10.3 Future Work

The analysis conducted in this paper to measure the orphan exclusion attack assumes that no natural orphan blocks occur, meaning honest blocks do not orphan each other. However, in practice, network delays in block propagation can cause non-adversarial blocks to orphan one another, making the orphan exclusion attack more destructive. Prior works such as [14, 15] model network delays by discounting honest mining power. Applying the same approach allows us to assess system security under such delays. Nonetheless, as shown in [15] and evidenced by the minimal number of orphaned blocks in recent years [3], Bitcoin's long block interval reduces the likelihood of honest blocks orphaning each other, suggesting that delays should not significantly extend the impact of the orphan exclusion attack. However, an interesting future research direction is to analyze the severity of the orphan exclusion attack combined with network partitioning and an eclipse attack [15, 28]. This partitioning can lead to honest blocks orphaning each other, amplifying the attack's impact.

Our paper analyzed the profitability of selfish mining under period-based DAMs, where the difficulty is adjusted at the end of a fixed period [29]. However, other types of DAMs exist, such as sliding window-based DAMs that can be modified to incorporate orphan blocks. The orphan exclusion attack introduced

in this paper is not limited to period-based DAMs and can also be applied to sliding window-based DAMs. As future work, one could analyze selfish mining profitability under a sliding window DAM that accounts for orphan blocks.

Another potential direction would be to use the Bitcoin backbone model introduced in [14] to demonstrate that, under the modified DAM and uncle block definition, Bitcoin can still satisfy the common-prefix and chain-quality properties.

6.11 Conclusion

While it was widely believed that incorporating orphan block counts into Bitcoin's difficulty adjustment mechanism (DAM) would render selfish mining unprofitable, this paper disproves that belief by introducing two new attacks: the smart intermittent selfish mining attack and the orphan exclusion attack. These attacks demonstrated that selfish mining remains more profitable than the honest strategy, even when orphan blocks are considered in the DAM. In this paper, we analyzed the profitability of various selfish mining strategies under different DAMs. Additionally, using probability analysis and combinatorial tools, we assessed the impact of the orphan exclusion attack on selfish mining profitability. Our findings indicated that while including orphan blocks in Bitcoin's DAM cannot entirely eliminate selfish mining profitability, it can limit the selfish miner's gains to an acceptable level, making orphan reporting a viable countermeasure.

Bibliography

[1] Bitcoin - average mining costs. https://en.macromicro.me/charts/29435/bitcoin-production-total-cost.

[2] Selfish mining idea. https://bitcointalk.org/index.php?topic=2227.msg29606#msg29606, 2010.

[3] Number of bitcoin orphan blocks. https://www.blockchain.com/explorer/charts/n-orphaned-blocks, 2024.

[4] Hamid Azimy, Ali A Ghorbani, and Ebrahim Bagheri. Preventing proof-of-work mining attacks. *Information Sciences*, 608:1503–1523, 2022.

[5] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE symposium on security and privacy*, pages 104–121. IEEE, 2015.

[6] Jonah Brown-Cohen, Arvind Narayanan, Alexandros Psomas, and S Matthew Weinberg. Formal barriers to longest-chain proof-of-stake protocols. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 459–473, 2019.

[7] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 3(37), 2014.

[8] Amir Dembo, Sreeram Kannan, Ertem Nusret Tas, David Tse, Pramod Viswanath, Xuechao Wang, and Ofer Zeitouni. Everything is a race and nakamoto always wins. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 859–878, 2020.

[9] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*, pages 436–454. Springer, 2014.

[10] Chen Feng and Jianyu Niu. Selfish mining in ethereum. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 1306–1316. IEEE, 2019.

[11] Matheus VX Ferreira and S Matthew Weinberg. Proof-of-stake mining games with perfect randomness. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 433–453, 2021.

[12] Daniel Fullmer and A Stephen Morse. Analysis of difficulty control in bitcoin and proof-of-work blockchains. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 5988–5992. IEEE, 2018.

[13] Paul A Gagniuc. *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017.

[14] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. In *Annual International Cryptology Conference*, pages 291–323. Springer, 2017.

[15] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 3–16, 2016.

[16] Guy Goren and Alexander Spiegelman. Mind the mining. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 475–487, 2019.

[17] Cyril Grunspan and Ricardo Pérez-Marco. On profitability of selfish mining. *arXiv preprint arXiv:1805.08281*, 2018.

[18] Cyril Grunspan and Ricardo Pérez-Marco. Profit lag and alternate network mining. In *The International Conference on Mathematical Research for Blockchain Economy*, pages 115–132. Springer, 2023.

[19] Ethan Heilman. One weird trick to stop selfish miners: Fresh bitcoins, a solution for the honest miner. In *International Conference on Financial Cryptography and Data Security*, pages 161–162. Springer, 2014.

[20] Dragos I Ilie, Sam M Werner, Iain D Stewart, and William J Knottenbelt. Unstable throughput: when the difficulty algorithm breaks. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–5. IEEE, 2021.

[21] Aggelos Kiayias, Nikos Leonardos, and Dionysis Zindros. Mining in logarithmic space. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3487–3501, 2021.

[22] Yujin Kwon, Hyoungshick Kim, Jinwoo Shin, and Yongdae Kim. Bitcoin vs. bitcoin cash: Coexistence or downfall of bitcoin cash? In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 935–951. IEEE, 2019.

[23] Jihye Lee and Yoonjeong Kim. Preventing bitcoin selfish mining using transaction creation time. In *2018 International Conference on Software Security and Assurance (ICSSA)*, pages 19–24. IEEE, 2018.

[24] Tao Li, Zhaojie Wang, Guoyu Yang, Yang Cui, Yuling Chen, and Xiaomei Yu. Semi-selfish mining based on hidden markov decision process. *International Journal of Intelligent Systems*, 36(7):3596–3612, 2021.

[25] Nadisha Madhushanie, Sugandima Vidanagamachchi, and Nalin Arachchilage. Selfish mining attack in blockchain: a systematic literature review. *International Journal of Information Security*, pages 1–19, 2024.

[26] Michael Mirkin, Yan Ji, Jonathan Pang, Ariah Klages-Mundt, Ittay Eyal, and Ari Juels. Bdos: Blockchain denial-of-service. In *Proceedings of the 2020 ACM SIGSAC conference on Computer and Communications Security*, pages 601–619, 2020.

[27] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[28] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 305–320. IEEE, 2016.

[29] Kevin Alarcón Negy, Peter R Rizun, and Emin Gün Sirer. Selfish mining re-examined. In *International Conference on Financial Cryptography and Data Security*, pages 61–78. Springer, 2020.

[30] Kervins Nicolas, Yi Wang, and George C Giakos. Comprehensive overview of selfish mining and double spending attack countermeasures. In *2019 IEEE 40th Sarnoff Symposium*, pages 1–6. IEEE, 2019.

[31] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *Proceedings of the ACM symposium on principles of distributed computing*, pages 315–324, 2017.

[32] Saha Reno and Simu Sultana. Preventing selfish mining in public blockchain using alarming block and block interval time approach. In *2022 International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*, pages 988–993. IEEE, 2022.

[33] Fabian Ritz and Alf Zugenmaier. The impact of uncle rewards on selfish mining in ethereum. In *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 50–57. IEEE, 2018.

[34] Muhammad Saad, Laurent Njilla, Charles Kamhoua, and Aziz Mohaisen. Countering selfish mining in blockchains. In *2019 International Conference on Computing, Networking and Communications (ICNC)*, pages 360–364. IEEE, 2019.

[35] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 515–532. Springer, 2016.

[36] Roozbeh Sarenche, Svetla Nikova, and Bart Preneel. Deep selfish proposing in longest-chain proof-of-stake protocols. *Financial Cryptography and Data Security*, 2024.

[37] Roozbeh Sarenche, Ren Zhang, Svetla Nikova, and Bart Preneel. Time-averaged analysis of selfish mining in bitcoin. Cryptology ePrint Archive, Paper 2024/363, 2024.

[38] Siamak Solat and Maria Potop-Butucaru. Brief announcement: Zeroblock: Timestamp-free prevention of block-withholding attack in bitcoin. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 356–360. Springer, 2017.

[39] Taotao Wang, Soung Chang Liew, and Shengli Zhang. When blockchain meets ai: Optimal mining strategy achieved by machine learning. *International Journal of Intelligent Systems*, 36(5):2183–2207, 2021.

[40] Wenbo Wang, Dinh Thai Hoang, Peizhao Hu, Zehui Xiong, Dusit Niyato, Ping Wang, Yonggang Wen, and Dong In Kim. A survey on consensus mechanisms and mining strategy management in blockchain networks. *Ieee Access*, 7:22328–22370, 2019.

[41] Ren Zhang and Bart Preneel. Publish or perish: A backward-compatible defense against selfish mining in bitcoin. In *Cryptographers' Track at the RSA Conference*, pages 277–292. Springer, 2017.

[42] Ren Zhang and Bart Preneel. Lay down the common metrics: Evaluating proof-of-work consensus protocols' security. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 175–192. IEEE, 2019.

[43] Chencheng Zhou, Liudong Xing, Qisi Liu, and Honggang Wang. Effective selfish mining defense strategies to improve bitcoin dependability. *Applied Sciences*, 13(1):422, 2022.

[44] Roi Bar Zur, Ittay Eyal, and Aviv Tamar. Efficient mdp analysis for selfish-mining in blockchains. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 113–131, 2020.

Chapter 7

Mining Power Destruction Attacks in the Presence of Petty-Compliant Mining Pools

Publication Data

Roozbeh Sarenche, Svetla Nikova, and Bart Preneel, "Mining Power Destruction Attacks in the Presence of Petty-Compliant Mining Pools," to appear in *International Conference on Financial Cryptography and Data Security (FC)*, 2025 (https://arxiv.org/abs/2502.07410).

Contributions

I was the main author involved in Conceptualization, Methodology, Writing Original Draft, Software, and Visualization.

Abstract. Bitcoin's security relies on its Proof-of-Work consensus, where miners solve puzzles to propose blocks. The puzzle's difficulty is set by the difficulty adjustment mechanism (DAM), based on the network's available mining power. Attacks that destroy some portion of mining power can exploit the DAM to lower difficulty, making such attacks profitable. In this paper, we analyze three types of mining power destruction attacks in the presence of petty-compliant mining pools: selfish mining, bribery, and mining power distraction attacks. We analyze selfish mining while accounting for the distribution of mining power among pools, a factor often overlooked in the literature. Our findings indicate that selfish mining can be more destructive when the non-adversarial mining share is well distributed among pools. We also introduce a novel bribery attack, where the adversarial pool bribes petty-compliant pools to orphan others' blocks. For small pools, we demonstrate that the bribery attack can dominate strategies such as selfish mining or undercutting. Lastly, we present the mining distraction attack, where the adversarial pool incentivizes petty-compliant pools to abandon Bitcoin's puzzle and mine for a simpler puzzle, thus wasting some part of their mining power. Similar to the previous attacks, this attack can lower the mining difficulty, but with the difference that it does not generate any evidence of mining power destruction, such as orphan blocks.

7.1 Introduction

Bitcoin [33] was the starting point of the exciting journey into the blockchain and cryptocurrency era. Despite the introduction of many new cryptocurrencies, Bitcoin has maintained the highest market capitalization [1] and remains the most famous cryptocurrency. Any serious attack that threatens Bitcoin's progress not only harms Bitcoin users but also impacts users of other cryptocurrencies and blockchain platforms. Although Bitcoin has operated smoothly since its emergence, the absence of a serious attack thus far does not guarantee that its security will not be compromised in the future. Since the introduction of Bitcoin, numerous research papers [5, 16, 17] have analyzed its security and explored potential attacks [6, 14, 28, 38] that could target it. Identifying potential flaws in Bitcoin's underlying mechanism and studying possible solutions benefit Bitcoin's progress while serving as inspiration for new cryptocurrencies.

Several attacks have been introduced in the literature aimed at destroying the efforts of other miners in producing valid blocks, which we refer to as *mining power destruction attacks*. Examples include selfish mining [14, 38], block withholding [13, 37], power adjusting withholding and bribery selfish mining [15], block denial of service (BDoS) [31], undercutting [11], eclipse attack [34], Pitchforks [23], and script puzzle distraction attack [43]. While

these attacks differ in their specific methods, they all share the common goal of destroying the mining power of other miners. The intuition behind the profitability of these attacks is the principle that mining rewards are distributed among the participating mining powers in proportion to their contributions to the chain extension. By destroying another miner's share, the adversary increases its own relative contributions, which consequently boosts its revenue. No mining power destruction attacks can be more profitable than following the honest strategy during the initial difficulty epoch [19, 41]. The adversary must wait for the subsequent difficulty adjustment mechanism to observe the impact of these attacks on its revenue. In contrast to attacks such as double spending, where the adversary receives revenue immediately upon a successful attack, mining power destruction attacks must be continued for at least two weeks (the length of one epoch) to become profitable.

In this paper, we analyze mining power destruction attacks in Bitcoin within the context of *petty-compliant mining pools*, referred to as a semi-rational setting. The analysis of these attacks in a semi-rational setting differs in two important ways from their analysis in the presence of altruistic mining pools. First, in semi-rational settings, we cannot naively assume that mining pools will always follow honest behavior, especially when they are the victims of an attack. As petty-compliant pools, they may choose to deviate from the honest strategy to defend against mining power destruction attacks and protect their mining efforts. Second, an adversarial mining pool in such settings may carry out incentive manipulation attacks [25], such as bribery, to convince non-victim petty-compliant mining pools to adopt its desired strategy, thereby increasing the success likelihood for its mining power destruction attacks. This potential for incentive manipulation allows adversarial mining pools even with a limited share of mining power to effectively attack the network. In this paper, we present a selfish mining analysis from a new perspective and introduce two novel mining power destruction attacks: bribery and mining power distraction attacks.

Selfish mining attack (Sect. 7.3): Selfish mining, introduced by Eyal et al. [14], is a well-studied attack in the context of Bitcoin. Numerous papers [8, 14, 19, 38, 40, 50] have analyzed the impact of selfish mining on the adversarial block ratio and profitability. However, most of these studies assume that, apart from the adversarial mining nodes, all the remaining mining nodes in the network follow the honest fork choice rule, meaning they always mine on the longest chain or the earliest of the longest forks. Carlsten et al. [11] introduced the concept of petty-compliant mining nodes, highlighting that, in a fork race, these mining nodes choose the fork that offers the highest return.[1] To the best of our

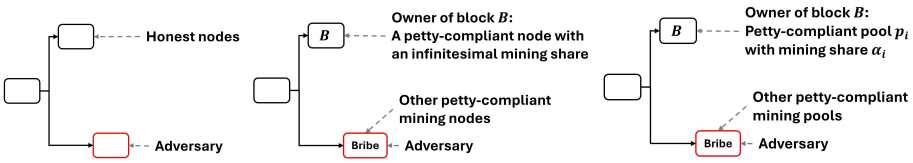
[1] [11] uses the concept of petty-compliant nodes in the undercutting attack analysis, overlooking them in the selfish mining analysis.

knowledge, the first analysis of selfish mining in the presence of petty-compliant mining nodes was conducted by Bar-Zur et al. [9]. They showed that in a semi-rational setting, the mining share threshold for a profitable deviation from the honest strategy is reduced.

Despite considering petty-compliant mining nodes, the analysis in [9] is based on the implicit assumption that each mining node controls only an infinitesimal share of the total mining power, thereby overlooking the presence of any petty-compliant *mining pools*. A mining pool can be viewed as a group of infinitesimal mining nodes working together. Assuming no petty-compliant mining pools exist, the analysis can simply presume that, during a fork race, if the adversary places a bribe on top of the adversarial fork, almost all the network's mining power will mine on top of the adversarial fork. This is because the mining nodes that do not adopt the adversarial fork are only those that own a block in the non-adversarial fork, with a total mining share that is infinitesimal. However, when considering the existence of mining pools, the presence of a bribe offered by the adversary does not guarantee that all the network's mining power will mine on top of the adversarial fork. If a mining pool has already mined a block in the non-adversarial fork, it will continue to mine on top of that fork with its entire mining power, which can no longer be assumed to be infinitesimal. Figure 7.1 intuitively illustrates the mining share distributions between forks in a fork race, in the altruistic setting, in the presence of infinitesimally small petty-compliant mining nodes, and in the presence of petty-compliant mining pools.

In this paper, we analyze selfish mining in the presence of petty-compliant mining pools and examine the effect of mining power decentralization on selfish mining profitability. Our theoretical and Markov Decision Process (MDP)-based analysis shows that selfish mining is more destructive in settings where the non-adversarial mining pools are more decentralized. This suggests that the greater the gap in mining power share between the adversarial mining pool and the other pools, the more profitable selfish mining becomes.

Bribery attack (Sect. 7.4): In the blockchain literature, various bribery attacks have been introduced for different purposes. These attacks can range from attempts to censor a single transaction, as seen in [32, 44], to efforts aimed at rewriting the history of blockchain blocks to facilitate a successful double-spending attack [10, 24, 28, 30]. The former requires a relatively small bribe to incentivize miners not to include a specific transaction, while the latter necessitates a substantial budget to persuade miners to abandon the longest chain and mine on top of a block that is deep within the chain. In this paper, we introduce a novel bribery attack aimed at destroying mining hash power. In our attack, the adversarial mining pool bribes petty-compliant mining pools to orphan the blocks of other mining pools, with the sole intention of wasting



(a) Altruistic setting: $\alpha_{\mathcal{A}}^{\texttt{fork}} = \alpha_{\mathcal{A}}$.

(b) Semi-rational setting with infinitesimal mining nodes: $\alpha_{\mathcal{A}}^{\texttt{fork}} \approx 1$.

(c) Semi-rational setting with mining pools: $\alpha_{\mathcal{A}}^{\texttt{fork}} = 1 - \alpha_i$.

Figure 7.1: Mining share distribution between forks in a block race. The adversarial block, denoted in red, is published later than the rival block in the block race. We denote by $\alpha_{\mathcal{A}}$, α_i, and $\alpha_{\mathcal{A}}^{\texttt{fork}}$ the mining shares of the adversarial mining pool, mining pool p_i, and all mining nodes extending the adversarial fork, respectively.

a portion of their mining power. Orphaning these blocks can reduce mining difficulty, resulting in revenue that compensates for the bribe paid. Our bribery attack is similar to selfish mining, except that instead of gambling on its own block to orphan another, the adversary pays a bribe less than the block reward to orphan a block. The advantage of the bribery attack over attacks such as selfish mining and feather forking is that it is risk-free for the adversarial pool, meaning the attacker does not risk losing its block or bribing budget. Additionally, a mining pool with any arbitrary share can always exploit other pools with a lesser share, making the attack profitable even for small mining pools.

Distraction attack (App. 7.A5): As discussed in the literature [19, 41], modifying the Bitcoin Difficulty Adjustment Mechanism (DAM) to adjust the difficulty based on the total active mining power—both wasted and effective—can mitigate mining power destruction attacks that generate valid proof of mining power destruction. For instance, selfish mining and the introduced bribery attack become unprofitable under such a modified DAM, as the orphan blocks generated during these attacks serve as evidence of mining power destruction. However, there are other types of mining strategies and attacks discussed in the literature, such as smart mining [18], coin hopping [20, 22, 26], and distraction attacks [43], which can destroy mining power without leaving evidence of destruction. In smart mining, a miner alternates between being idle and mining honestly, while in coin hopping, the miner switches between mining on different networks. Both strategies aim to manipulate and reduce the mining difficulty level. In distraction attacks, the adversary incentivizes miners to stop mining the Bitcoin puzzle and instead mine an alternative one. An example of a distraction attack is the script puzzle [43], where the adversary bribes miners through a smart contract-based puzzle to divert them from the Bitcoin chain. The goal of

the script puzzle attack is to facilitate a double-spending attack or gain majority control of the network, which requires both a significant share of mining power and a substantial bribe (equivalent to six Bitcoin block rewards) for successful execution. These strategies and attacks can target Bitcoin without leaving any evidence, making them difficult to mitigate without relying on external trusted platforms.

In this paper, we introduce a novel distraction attack that aims to destroy a portion of mining power without leaving any evidence of power destruction. In our attack, the adversarial mining pool publishes a Proof-of-Work (PoW) puzzle with lower difficulty on another platform and incentivizes petty-compliant mining pools to mine the lower-difficulty puzzle. This attack can be carried out with minimal bribes, less than the value of a block reward.

7.2 Preliminaries and System Model

In our system model, time is divided into smaller units referred to as rounds. We use λ to denote the block mining rate of the network, representing the number of blocks mined per unit of time. We denote by $\textsc{Rev}_i(t;\pi)$ and $\textsc{Cost}_i(t;\pi)$ the revenue and the cost of mining pool p_i in round t under strategy π, respectively. If all the mining pools follow the honest strategy, the average per-round revenue of the mining pool p_i with mining share α_i is equal to $\alpha_i \cdot \lambda R$, where R denotes the block reward. If mining pool p_i mines with its whole mining power, its average mining cost per round is equal to $\alpha_i \cdot c_i$, where c_i denotes the average normalized mining cost of pool p_i per round.

Definition 7.2.1 (Time-averaged profit)**.** The time-averaged profit (per-round profit) of pool p_i following strategy π is defined as follows:

$$\begin{aligned} \textsc{Profit}_i^t(\pi) &= \frac{\sum_{t'=0}^{t-1} \Big(\textsc{Rev}_i(t';\pi) - \textsc{Cost}_i(t';\pi) \Big)}{t}, \\ \textsc{Profit}_i(\pi) &= \lim_{t\to\infty} \textsc{Profit}_i^t(\pi) \ . \end{aligned} \tag{7.1}$$

We define honest, petty-compliant, and adversarial mining pools as follows.

Definition 7.2.2 (Honest mining pool)**.** An honest mining pool is defined as a mining pool that i) always chooses the longest chain available in its view (in case of a tie, it chooses the block that was seen first) as its canonical chain to mine on top of, and ii) once it mines a new block, it immediately publishes the block to all other mining pools.

To define petty-compliant mining pools, we first define the chain expected return and semi-rational fork choice rule.

Definition 7.2.3 (Chain expected return)**.** Let π_C and r_C denote the strategy of mining on top of a given chain C, and its expected return, respectively. The expected return r_C is defined as follows:

$$r_C = \sum_{t=0}^{\infty} \gamma^t \text{REV}_i(t; \pi_C) \ , \tag{7.2}$$

where γ is a decaying factor set by each mining pool.[2]

For a given chain C, we denote by $|C|$ the length of the chain.

Definition 7.2.4 (Semi-rational fork choice rule)**.** Let $C^{\texttt{long}}$ denote the longest chain available in the view of petty-compliant mining pool p_i (if there are multiple chains of equal length, consider the one that was seen first by the mining pool). Also, let $\texttt{CH}^D$ denote the set of all the chains C in the view of petty-compliant mining pool p_i that satisfy $|C^{\texttt{long}}| - |C| \leq D$. According to the (ϵ, D)-semi-rational fork choice rule, if there is a chain $C^* \in \texttt{CH}^D$ that satisfies the following conditions:

1. $r_{C^*} \geq r_C$ for all $C \in \texttt{CH}^D$, and
2. $r_{C^*} - r_{C^{\texttt{long}}} > \epsilon \alpha_i R$,

then C^* is the canonical chain. Otherwise, $C^{\texttt{long}}$ is the canonical chain.

We refer to ϵ as the incentivizing factor, representing the threshold of normalized loss a mining pool tolerates before deviating from the honest strategy. A discussion of the parameters ϵ and D, as well as their role in the semi-rational fork choice definition is provided in App. 7.A1.

Definition 7.2.5 (Petty-compliant mining pool)**.** An (ϵ, D)-petty-compliant mining pool is defined as a mining pool that i) follows the (ϵ, D)-semi-rational fork choice rule to select the canonical chain to mine on top of, and ii) once it mines a new block, it immediately publishes the block to all other mining pools unless it is incentivized not to do so.

In this paper, petty-compliant pools are restricted to specific strategies defined for each attack and cannot act arbitrarily.

Definition 7.2.6 (Adversarial mining pool)**.** The adversarial mining pool may arbitrarily deviate from the honest strategy (for example, by delaying the publication of its blocks) or execute an incentive manipulation attack to induce petty-compliant mining pools to deviate from the honest strategy.

[2]The presence of γ in the expected return definition makes it subjective to each mining pool's view of profitability. This reflects reality, as some pools prefer immediate rewards (low γ), while others prioritize long-term profit (with γ close to 1). In our paper analysis, we assume γ is close to 1 and that petty-compliant pools do not account for the impact of difficulty adjustment on the chain expected return.

Definition 7.2.7 (Semi-rational environment)**.** An (ϵ, D)-semi-rational environment is an environment in which any non-adversarial mining pool is an (ϵ', D)-petty-compliant mining pool, where $\epsilon' \leq \epsilon$. If $D = \infty$, we simply denote the environment as the ϵ-semi-rational environment.

System model. We assume our model operates within an (ϵ, D)-semi-rational environment. The system comprises a set of petty-compliant mining pools, denoted by p_i for i in $1, 2, \cdots, N$, alongside an adversarial mining pool denoted by $p_{\mathcal{A}}$. We denote by α_i and $\alpha_{\mathcal{A}}$ the mining power share of the ith petty-compliant mining pool and the adversarial mining pool, respectively, where $\alpha_{\mathcal{A}} + \sum_{i=1}^{N} \alpha_i = 1$. Our model assumes a fixed block reward of R; however, the adversarial mining pool may also place a bribe on top of any block. A discussion of the limitations of our system model is provided in App. 7.A2.

Definition 7.2.8 (Normalized bribe)**.** The normalized bribe `br` is defined as the amount of bribe divided by the block reward R.

According to Def. 7.2.8, a normalized bribe of `br` implies that the total amount of the bribe is equal to `br`R.

7.3 Selfish Mining Attack

In this section, we analyze selfish mining in the presence of petty-compliant mining pools. Selfish mining results in a fork race between the adversarial and non-adversarial forks, with only one eventually being included in the canonical chain. The presence of petty-compliant mining pools offers some benefits to an adversarial pool. During a fork race, the adversary can bribe these pools to mine on the adversarial fork, boosting its chances of winning. The adversary can use in-band methods such as whale transactions [28] (discussed in App. 7.A3.1) or out-of-band methods such as smart contracts [30, 45] to bribe miners. It may also leave transaction fees in mempool as bribes for others [9, 11].

Analyzing selfish mining in the presence of petty-compliant mining pools requires revisiting the selfish mining analysis typically applied to altruistic settings or semi-rational settings with infinitely many infinitesimal mining nodes. Under the altruistic assumption, the adversarial mining pool is guaranteed to win the fork race if its fork is longer than the competing fork. Additionally, if the adversarial pool propagates its fork faster, it increases its chances of winning a same-height fork race. In semi-rational settings with infinitely many infinitesimal mining nodes, the analysis may assume that the adversary can incentivize **all** petty-compliant nodes to abandon the non-adversarial fork with a bribe, as the mining share of infinitesimal miners in the non-adversarial fork is negligible.

However, the scenarios that can actually occur when selfish mining takes place in practice, particularly in the presence of petty-compliant mining pools, may differ significantly from the analyses conducted under the simplified assumptions of altruistic and infinitesimal miners. In the presence of petty-compliant mining pools, the adversary is not necessarily guaranteed to win the fork race solely based on the length of its fork, the speed at which it propagates its fork, or even the minimal bribe placed on its fork. Petty-compliant mining pools select the fork to mine based on its expected return, where the longest, fastest-propagated, or bribed chain may not always offer the highest expected return. For instance, consider a fork race between the non-adversarial fork with length $l_{\overline{\mathcal{A}}}$ and the adversarial fork with length $l_{\mathcal{A}} > l_{\overline{\mathcal{A}}}$. If the petty-compliant mining pool p_i has $n > 0$ blocks in the non-adversarial fork, although the non-adversarial fork is shorter than the adversarial fork, p_i may still be incentivized to continue mining on top of the non-adversarial fork to revive its n blocks included in it. This example suggests that in the presence of petty-compliant mining pools, the strategy employed by the selfish pool and its profitability differ from those in altruistic settings or in environments with infinitely many infinitesimal nodes.

7.3.1 Theoretical Analysis

To analyze the selfish mining attack in the presence of petty-compliant mining pools, we first introduce metrics to assess the distribution of mining power among these pools.

Definition 7.3.1 (Centralization factor)**.** Let $\mathcal{P} = \{p_1, p_2, \cdots, p_N\}$ denote the set of all mining pools available in the environment. Also, let α_{p_i} denote the corresponding mining power share of mining pool $p_i \in \mathcal{P}$. The centralization factor, which is denoted by β, is defined as follows:

$$\beta = \sum_{p_i \in \mathcal{P}} {\alpha_{p_i}}^2 \ . \tag{7.3}$$

The centralization factor β can take on values in the range of $(0, 1)$, with higher β values indicating a more centralized network. In a fully decentralized network, β approaches 0. Within a network where the maximum mining share of its mining pools is denoted by α, the centralization factor β is less than or equal to α, with the upper bound case of $\beta = \alpha$ occurring when all mining pools possess a mining share exactly equal to α.

Definition 7.3.2 (Residual centralization factor, Pool advantage)**.** Let $\mathcal{P}_{\overline{i}} = \{p_1, p_2, \cdots, p_N\} \setminus \{p_i\}$ denote the set of all mining pools excluding mining pool p_i whose mining power share is denoted by α_{p_i}. Also, let α_{p_j} denote the corresponding mining power share of mining pool $p_j \in \mathcal{P}_{\overline{i}}$. The residual

centralization factor w.r.t. mining pool p_i, which is denoted by β_i, is defined as follows:

$$\beta_i = \frac{\sum_{p_j \in \mathcal{P}_{\frac{-}{i}}} {\alpha_{p_j}}^2}{1 - \alpha_{p_i}} \ . \tag{7.4}$$

The mining advantage of pool p_i is defined to be $1 - \beta_i$.

In the following, we demonstrate that a mining pool with a lower residual centralization factor (i.e., higher mining advantage) has a higher chance of conducting a successful selfish mining attack. To get the intuition of why pool advantage is defined as above, we first review the following lemma.

Lemma 7.1. Consider a fork race within an ϵ-semi-rational environment, where the length of both semi-rational and adversarial forks is equal to 1, and a normalized bribe (see Def. 7.2.8) of $\texttt{br} = \epsilon$ is available on top of the adversarial fork. The probability of the event that the next block is mined on top of the adversarial fork is equal to the mining advantage of the adversarial mining pool.

The proof of Lemma 7.1 is presented in App. 7.A3.2. In Def. 7.A3.1 presented in App. 7.A3.3, we introduce a simple selfish mining strategy π^{selfish} suitable for an $(\epsilon, D = 1)$-semi-rational environment. As already discussed, strategies designed for an altruistic environment cannot be easily applied in a semi-rational setting. The outcome of each action that the adversarial mining pool takes in the semi-rational environment must be evaluated from the perspective of all petty-compliant mining pools. The following theorem examines the effect of the adversarial centralization factor on the profitability of strategy π^{selfish}.

Theorem 7.1. Assume an $(\epsilon, D = 1)$-semi-rational environment[3] in which the mining share of each pool, excluding the adversarial pool, is less than 0.4302. In this environment, the selfish mining strategy π^{selfish} for an adversarial mining pool with mining share $\alpha_{\mathcal{A}}$ and residual centralization factor $\beta_{\mathcal{A}}$ can dominate the honest mining if the following inequality holds:

$$\beta_{\mathcal{A}} < \frac{\alpha_{\mathcal{A}} - \epsilon(1 - \alpha_{\mathcal{A}})^2}{(1 - \alpha_{\mathcal{A}})(1 - \epsilon)} \ . \tag{7.5}$$

The proof of Thm. 7.1 is presented in App. 7.A3.5. The assumption that the mining share of each non-adversarial mining pool is less than 0.4302 ensures that if an $(\epsilon, D = 1)$-petty-compliant mining pool has a single block in a fork that lags behind the longest fork by one block, it is incentivized to abandon its fork and adopt the longest fork (Lemma 7.2).

[3] $D = 1$ implies that the best chain in a mining pool view is a chain that is at most one block shorter than the longest chain.

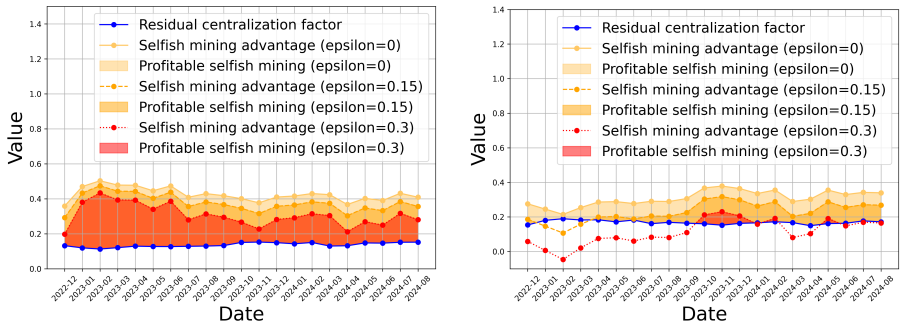
Note that strategy π^{selfish} is not the optimal selfish mining strategy that an adversarial mining pool can follow in an $(\epsilon, D = 1)$-semi-rational environment. The main goal of Thm. 7.1 is to show that a mining pool with a lower residual centralization factor has a higher chance of successfully executing a selfish mining attack. This implies that the profitability of selfish mining in a semi-rational environment depends not only on the mining share of the adversarial pool but also on the distribution of mining power among the remaining mining pools.

Corollary 7.1. Let P_1 and P_2 denote the first and second mining pools with the highest mining share in the network, whose corresponding mining shares are denoted by α_1 and α_2, respectively. Then, we can obtain the following statements for an ϵ-semi-rational environment:

- For $\epsilon > 0$, selfish mining dominate honest mining for mining pool P_1 if $\frac{\alpha_1}{1-\alpha_1} > \alpha_2 + \epsilon(1 - \alpha_1 - \alpha_2)$.
- For $\epsilon = 0$, selfish mining always dominate honest mining for mining pool P_1.

The proof of Cor. 7.1 is presented in App. 7.A3.6. According to Cor. 7.1, selfish mining is the dominant strategy for the largest mining pool if the network incentivizing threshold ϵ is less than the difference between the mining shares of the largest and the second largest mining pools. An important consideration in the literature on selfish mining is the minimum threshold of mining power required for a profitable attack. Perhaps the most well-known example is that, in an altruistic setting, a miner with a normal communication capability needs at least 25% of the network's mining power to successfully execute selfish mining [14]. These thresholds might suggest that if no mining pool holds a share above the threshold, selfish mining cannot threaten the network. However, Thm. 7.1 and Cor. 7.1 show that in practice, where multiple mining pools aim to maximize their payoffs, mining share alone is not the only factor determining the threat of selfish mining. The distribution of mining power among pools and the gap between their shares also play a critical role. Even if all mining pools hold less mining power than these well-known thresholds, a pool with a sufficiently large gap between its share and that of others can still find selfish mining profitable.

We define the ϵ-selfish mining advantage of the adversary as the right-hand side of inequality (7.5). According to Thm. 7.1, selfish mining dominates honest mining if the adversary's ϵ-selfish mining advantage exceeds its residual centralization factor. The mining power distribution for the first 8 months of 2024 is presented in Table 7.1 in App. 7.A3.7 [3]. Figure 7.2 depicts both the centralization factor and the ϵ-selfish mining advantage for the largest and second-largest mining pools over the months from December 2022 to August 2024. To generate Fig. 7.2, we used data on the number of blocks mined



(a) Largest mining pool

(b) Second largest mining pool

Figure 7.2: Periods of profitable selfish mining. According to Thm. 7.1, selfish mining becomes more profitable than honest mining when the selfish mining advantage of a mining pool (defined as the right-hand side of inequality (7.5)) surpasses its residual centralization factor (defined in Def. 7.3.2).

by the respective pools during these months to calculate their corresponding mining shares for each month. This figure illustrates in which months selfish mining could be considered profitable based on different values of the network incentivizing factor. As shown in Fig. 7.2a, selfish mining is always the dominant strategy for the largest pool, even with a high incentivizing factor of $\epsilon = 0.3$. If we assume $\epsilon = 0$, as can be seen in Fig. 7.2b, selfish mining is also the dominant strategy for the second-largest pool. However, as the incentivizing factor increases, some months are excluded from the profitable selfish mining period.

In App. 7.A3.8, we present our MDP-based implementation, available at [2], to derive the optimal selfish mining strategy in the semi-rational setting. In App. 7.A3.10, we discuss the duration of the initial financial loss associated with selfish mining and the cost incurred by the adversarial mining pool during this period.

7.4 Bribery Attack

Based on the selfish mining results from our MDP implementation presented in Table 7.4, the profitability of selfish mining drastically decreases as a pool's mining share decreases. The main reason for the marginal profitability of selfish mining for smaller pools is that, in selfish mining, an adversary is gambling on its own blocks, with no guarantee of winning the fork race. In other words,

when an adversary withholds a block to initiate a fork race, it either collects the block reward R or loses it entirely. For larger mining pools, the probability of winning the fork race is high, making selfish mining profitable. However, smaller pools face a significantly higher risk of losing the block reward in the fork race. As a result, small mining pools prefer not to initiate a fork race unless they know the competing mining pool in the fork race is also small.

Knowing that selfish mining is a marginally-profitable mining power destruction attack for smaller pools, this section introduces a novel bribery attack that (similar to selfish mining) aims to waste a portion of the network's hash power. However, unlike selfish mining, this bribery attack is risk-free for the adversary and is not limited to times when the adversary has mined a block, thereby allowing the adversary to increase its profitability. In this attack, an adversarial mining pool pays a bribe to any other mining pool that can orphan a specific block targeted by the adversarial pool. In other words, the adversarial pool incentivizes petty-compliant mining pools to orphan a block mined by another mining pool. The profitability of the bribery attack is based on the same principle as selfish mining. In both of these attacks, the adversary aims to waste part of the network's hash power to exploit the difficulty adjustment mechanism, thereby reducing mining difficulty. The key difference is that in selfish mining, the adversarial pool uses its own blocks to carry out the attack, whereas in the bribery attack, it uses its budget to incentivize other mining pools to conduct the attack. The bribery attack offers two key advantages over selfish mining. First, it is risk-free for the adversary, as the bribe is only paid to collaborating pools upon successfully orphaning a block. Second, it can be executed by pools with small mining shares, enabling them to exploit weaker pools.

Definition 7.4.1 (Target and rival blocks)**.** Let B_1 represent the head of the canonical chain that is mined on top of block B_0. Additionally, suppose an adversarial mining pool offers a bribe for publishing a block B_1' on top of B_0, where B_1' differs from B_1. In this case, B_1 and B_1' are referred to as the target block and the rival block of a bribery attack, respectively.

When an adversarial mining pool targets a block B_1 for a bribery attack, it is not guaranteed that a rival block will always be mined for the target block B_1. If no rival block is mined, the attack fails. However, if a rival block B_1' is mined for the target block B_1, the attack succeeds. In a successful bribery attack, both the target and rival blocks undergo a fork race. Note that the success of this bribery attack is independent of the outcome of the fork race, as neither of the blocks involved in the fork race is adversarial. Since only one block between the target and rival blocks can be included in the canonical chain, the orphaning of one block is definite, implying that the occurrence of the fork race is enough to consider the attack successful. The following theorem determines the maximum bribe the adversary can spend on the bribery attack while keeping it profitable.

Theorem 7.2. Consider an adversarial mining pool with a mining share $\alpha_{\mathcal{A}}$ that allocates a normalized bribe $\mathtt{br}$ for each target block B, which is payable only upon the successful mining of a rival block for block B. Assume k is the average number of target blocks per epoch for which a rival block is mined under the bribery attack. The time-averaged profit of the adversarial pool under the bribery attack exceeds the honest mining profit as long as $\mathtt{br} < \alpha_{\mathcal{A}}$, for any value of k.

The proof of Thm. 7.2 is presented in App. 7.A4.1. The idea behind the proof of Thm. 7.2 is to demonstrate that for each non-adversarial block that gets orphaned, the adversarial mining pool receives an additional revenue of $\alpha_{\mathcal{A}}R$ on average. Therefore, if the bribe spent on orphaning each non-adversarial block is less than $\alpha_{\mathcal{A}}R$, the adversarial mining pool can still earn a profit.

The question that arises is how the adversary can effectively use this bribe budget to incentivize other petty-compliant pools to mine a rival block for a target block. In the following section, we introduce a bribery attack and analyze it in the setting where all mining pools are aware of which pool has mined the target block B. Once a mining pool mines a block, it can reveal its identity in the coinbase transaction. According to the statistical information presented in [3], the ratio of unknown blocks in the first 8 months of 2024 is less than 8%, indicating that the majority of mining pools reveal their identity in their blocks. In App. 7.A4.2, we analyze the bribery attack in the setting where the miner of the target block is unknown.

7.4.1 Bribery Attack Under the Assumption of Known Miners

In this section, we present a smart-contract-based bribery attack under the assumption that the miners of target blocks are known. App. 7.A4.3 discusses a bribery attack using whale transactions.

7.4.1.1 Description of Bribery Attack Using Smart Contracts.

Let $\alpha_{\mathcal{A}}$ represent the mining share of the adversarial mining pool $p_{\mathcal{A}}$. Assume block B_1 denotes the head of the canonical chain and is mined by mining pool p_i with mining share α_i. From the perspective of the adversarial mining pool, block B_1 can be considered a valid target block for the smart contract-based bribery attack if the following conditions hold: 1) B_1 is a non-adversarial block, and 2) $\alpha_{\mathcal{A}} > \alpha_i + 2\epsilon$. If these conditions are not satisfied, the attack is not applicable, and the adversarial mining pool waits for the next block. If block B_1 is a valid target block, the adversarial mining pool proceeds with the bribery attack. Let B_0 denote the parent of the target block B_1. The adversarial mining pool deploys a smart contract and publishes it for all the mining pools.

The smart contract stores three parameters: the hash of block B_0 denoted by $\texttt{Hash}(B_0)$, the hash of block B_1 denoted by $\texttt{Hash}(B_1)$, and a difficulty target denoted by `Target` that is equal to the difficulty target of the epoch to which block B_1 belongs. The adversary deposits 2 normalized bribes $\texttt{br}_1 = \alpha_i + \epsilon$ and $\texttt{br}_2 = \epsilon$ in the smart contract. Anyone who submits a witness that proves the mining of a rival block $B_{\texttt{rival}}$ for block B_1 can withdraw $\texttt{br}_1$. A valid witness includes a Bitcoin `nonce`, a Merkle root `MR`, a Bitcoin coinbase transaction tx, a Merkle inclusion `proof`, and a payout address `add` in the host blockchain, and satisfies the conditions:

1. Demonstrates valid evidence of power destruction[4]:
   - A valid rival block is mined on top of block B_0: $\texttt{Hash}(B_{\texttt{rival}}) = \texttt{Hash}\big(\texttt{Hash}(B_0), \texttt{MR}, \texttt{nonce}\big) \leq \texttt{Target}$.
   - The rival block differs from the target block: $\texttt{Hash}(B_{\texttt{rival}}) \neq \texttt{Hash}(B_1)$.
2. Includes a payout address that belongs to the compliant miner:
   - The coinbase transaction is included in the list of block transactions, i.e., a valid Merkle inclusion `proof` is available that proves coinbase transaction tx is a leaf of a Merkle tree with Merkle root `MR`.
   - The payout address `add` is included in the coinbase transaction.

If the witness transaction satisfies the conditions above, bribe $\texttt{br}_1$ will be transferred from the smart contract deposit to the payout address `add`, and the rival block hash $\texttt{Hash}(B_{\text{rival}})$ gets stored in the contract as the hash of the rival block. Anyone who submits a witness that proves the mining of a valid supporting block on top of the rival block B_{rival} can withdraw $\texttt{br}_2$. The witness verification is similar to the previous case with the difference that the hash of the parent block should be equal to $\texttt{Hash}(B_{\text{rival}})$. Note that this smart contract should have a time lock, after the expiration of which the adversarial mining pool is able to withdraw the deposits. This is to prevent any loss if the attack fails. During this attack, the adversarial mining pool follows the following strategy: if no rival block is published, it mines on top of the target block. Once a rival block is published, it switches to mining on top of the rival block.

Theorem 7.3. In an ϵ-semi-rational environment, an adversarial mining pool with mining share $\alpha_{\mathcal{A}}$ is incentivized to conduct a smart contract-based bribery attack on any target block B if block B is mined by a mining pool with mining share α_i, where $\alpha_{\mathcal{A}} > \alpha_i + 2\epsilon$.

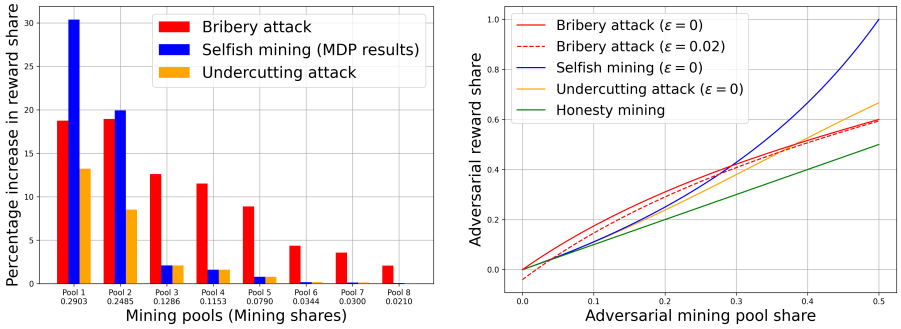
The proof of Thm. 7.3 is presented in App. 7.A4.4. The proof demonstrates that the adversarial mining pool must pay a normalized bribe of $\alpha_i + 2\epsilon$ to incentivize

[4]Verifying the transactions in the rival block is unnecessary. Any user who solves a new PoW puzzle (not necessarily a valid Bitcoin block) deserves the bribe.

petty-compliant pools to mine a rival block for a target block mined by a pool with share α_i. Each bribery attack orphans one non-adversarial block, granting the adversary additional normalized revenue of $\alpha_{\mathcal{A}}$. Thus, if this additional revenue $\alpha_{\mathcal{A}}$ exceeds the bribe $\alpha_i + 2\epsilon$, the attack becomes profitable.

According to Thm. 7.3, assuming $\epsilon = 0$, an adversarial mining pool can perform a bribery attack on blocks mined by any mining pool whose mining power is less than the adversarial pool's mining share. The bribery attack is similar to the undercutting attack introduced in [11]. The key difference is that in undercutting, the adversarial mining pool uses its own block to undercut a target block. However, in the bribery attack, the adversarial mining pool incentivizes other mining pools to undercut a target block. Note that the undercutting attack is a subset of the possible actions considered in our selfish mining MDP analyzer introduced in Sect. 7.A3.8. The Markov chain analysis of both the introduced bribery attack and the undercutting attack in the presence of petty-compliant pools is presented in App. 7.A4.5 and App. 7.A4.6, respectively.

Figure 7.3a, illustrates the percentage increase in reward share for the smart contract-based bribery attack, the selfish mining attack, and the undercutting attack, calculated for the 8 largest mining pools, based on the mining distribution in Table 7.1 and incentivizing factor $\epsilon = 0$. Figure 7.3b depicts the reward share that an adversarial mining pool can achieve in the presence of infinitesimal petty-compliant mining nodes. As can be seen in both Figures 7.3a and 7.3b, the bribery attack can dominate both selfish mining and undercutting in an ϵ-semi-rational environment when the adversarial mining pool's share is small. The higher profitability of the bribery attack can be attributed to two main factors. First, in the bribery attack, the adversarial mining pool does not risk losing its blocks. Second, the bribery attack is not limited to the times when the adversarial mining pool has mined a block. In the undercutting attack, once the tip of the blockchain is a valid target block, the adversarial pool must mine the next block to successfully undercut the target block. This probability is relatively low, especially if the adversarial mining pool is small. In contrast, in the bribery attack, the adversarial mining pool can incentivize all mining pools, except the target block miner, to undercut the target block. Compared to other bribery attacks in the literature [10, 24, 28, 30], where double-spent transactions compensate the adversary, in the introduced bribery attack, the protocol itself compensates the adversary, eliminating the need for performing deep block reorganizations. Additionally, the budget for each instance of the introduced bribery attack is only a fraction of a block reward, significantly less than the double-spending bribes, which require several block rewards.



(a) Petty-compliant mining pools (real-world mining power distribution).

(b) A single adversarial pool and infinitesimal petty-compliant nodes.

Figure 7.3: The adversarial reward share obtained from different attacks.

7.4.1.2 Duration and Cost of the Initial Loss Period.

Mining destruction attacks, including the proposed bribery attack, incur an initial period of financial loss before the mining difficulty adjusts. During the first epoch of the bribery attack, the adversary must pay bribes without any immediate increase in revenue per unit of time. This implies that the adversarial mining pool must allocate a budget to initiate the attack. In Fig. 7.4, we depict the normalized revenue advantage of the bribery attack for different real-world mining pools as a function of time. The revenue advantage is defined as the difference between the attack revenue and honest mining revenue since the launch of the attack [20]. As shown, during the first epoch of the attack—lasting longer than the standard 2-week epoch duration—the revenue advantage decreases, indicating a financial loss for the adversary. However, after the difficulty adjustment at the end of the first epoch, the revenue advantage of the bribery attack begins to increase and eventually becomes positive, indicating profitability.

7.5 Discussion

Thus far, Bitcoin has not experienced a serious mining destruction attack. This can be attributed to various factors, including the high required mining share and the long initial period of loss associated with these attacks. While these factors may limit mining power destruction attacks, they do not provide a 100% guarantee that Bitcoin will remain immune to such attacks in the future. For example, while a typical selfish mining attack requires a relatively high hash power (≥ 0.25) [38], conducting these attacks more strategically, such as

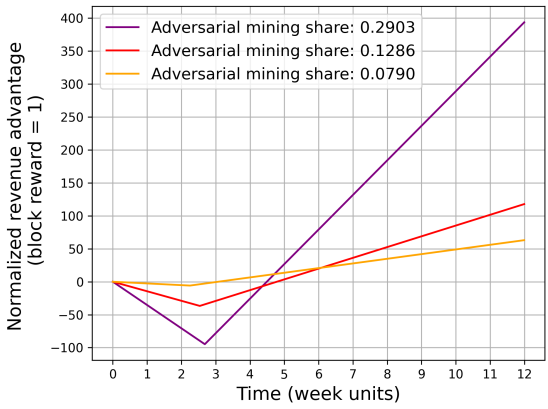


Figure 7.4: The normalized revenue advantage of the bribery attack over time.

through the bribery attack introduced in this paper, can make them feasible even for smaller mining pools. Additionally, as Bitcoin undergoes more halving events and transaction fees become an increasingly important source of rewards, mining destruction attacks can become more destructive. This shift toward a transaction-fee era can lower the mining share threshold for a profitable deviation from the honest strategy, as an adversarial mining pool may be incentivized to risk its low-fee-valuable blocks to orphan others' high-fee-valuable blocks. Orphaning such blocks returns their fee-valuable transactions to the mempool, giving the adversary an opportunity to claim them by mining the subsequent block. Eliminating the protocol reward can also remove the initial loss period for the adversary, as stolen fee-valuable transactions can offset the reduced block generation rate during the first epoch of the attack [11, 39]. This discussion highlights the importance of analyzing these threats and developing solutions to help Bitcoin remain secure against such attacks in the future.

Multiple solutions have been suggested in the literature, each with its own limitations. Some solutions rely on assessing the timeliness of block publication to detect honest and adversarial behavior [21, 27, 36, 42, 46]. However, beyond the limitations of these schemes in partially-synchronous and asynchronous networks, they cannot effectively mitigate attacks in the presence of rational mining pools. Note that rational miners choose the fork to mine on based on its return, regardless of whether the fork is honest or not. There are other solutions that suggest modifying the current Bitcoin difficulty adjustment mechanism (DAM) to prevent difficulty reduction during mining destruction attacks [4, 19, 41, 49]. These solutions rely on the existence of an honest minority that reports evidence of destruction attacks, such as orphaned blocks. This

evidence helps the DAM distinguish between an attack occurrence and a part of the network being offline, thereby preventing difficulty reduction during the attack. A comprehensive discussion on the impact of DAM on the profitability of selfish mining and bribery attacks, as well as modifications to the current Bitcoin DAM to mitigate these attacks, is presented in App. 7.A4.9. However, it is important to note that there exist certain mining power destruction attacks, such as the distraction attack introduced in this paper (App. 7.A5), that destroy mining power without generating any evidence. Furthermore, these solutions may limit adversarial profit at the cost of reducing the block generation rate, thereby penalizing honest miners. For more information on strategies to counter selfish mining, readers are referred to [29, 35, 41, 47].

7.6 Conclusion

This paper discussed how an adversarial mining pool can use various methods to destroy a portion of the network's mining power. To execute a mining power destruction attack, the adversarial mining pool must invest some resources, which can be viewed as an initial cost. These resources may include withholding a block in selfish mining or offering a bribe in a bribery attack. However, this initial cost can be compensated, as the destruction attack lowers mining difficulty, resulting in an increased block production rate for the adversarial mining pool. Victims of destruction attacks can take countermeasures to mitigate them. For example, hiding their identity as block miners is a simple yet effective strategy. Additionally, mining pools can collaborate on mutual agreements, such as dividing larger pools into smaller ones or adopting a more secure difficulty adjustment mechanism. However, for every wise solution, there may be an equally clever counterattack.

Bibliography

[1] Bitcoin market cap. https://www.binance.com/en/price/bitcoin, 2024.

[2] Our selfish mining MDP implementation. https://github.com/SelfishMiningPettyCompliant/MDP_analysis/tree/main, 2024.

[3] Pools' mining share. https://www.blockchain.com/explorer/charts/pools, 2024.

[4] Hamid Azimy, Ali A Ghorbani, and Ebrahim Bagheri. Preventing proof-of-work mining attacks. *Information Sciences*, 608:1503–1523, 2022.

[5] Christian Badertscher, Juan Garay, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. But why does it work? A rational protocol design treatment of Bitcoin. In *Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29-May 3, 2018 Proceedings, Part II 37*, pages 34–65. Springer, 2018.

[6] Samiran Bag, Sushmita Ruj, and Kouichi Sakurai. Bitcoin block withholding attack: Analysis and mitigation. *IEEE Transactions on Information Forensics and Security*, 12(8):1967–1978, 2016.

[7] Qianlan Bai, Yuedong Xu, Nianyi Liu, and Xin Wang. Blockchain mining with multiple selfish miners. *IEEE Transactions on Information Forensics and Security*, 18:3116–3131, 2023.

[8] Roi Bar-Zur, Ameer Abu-Hanna, Ittay Eyal, and Aviv Tamar. WeRLman: To tackle whale (transactions), go deep (RL). In *Proceedings of the 15th ACM International Conference on Systems and Storage*, pages 148–148, 2022.

[9] Roi Bar-Zur, Danielle Dori, Sharon Vardi, Ittay Eyal, and Aviv Tamar. Deep bribe: Predicting the rise of bribery in blockchain mining with deep

rl. In *2023 IEEE Security and Privacy Workshops (SPW)*, pages 29–37. IEEE, 2023.

[10] Joseph Bonneau. Why buy when you can rent? Bribery attacks on Bitcoin-style consensus. In *International Conference on Financial Cryptography and Data Security*, pages 19–26. Springer, 2016.

[11] Miles Carlsten, Harry Kalodner, S Matthew Weinberg, and Arvind Narayanan. On the instability of Bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 154–167, 2016.

[12] Hongwei Chen. Interesting series associated with central binomial coefficients, Catalan numbers and Harmonic numbers. *J. Integer Seq.*, 19(1):16–1, 2016.

[13] Ittay Eyal. The miner's dilemma. In *2015 IEEE symposium on security and privacy*, pages 89–103. IEEE, 2015.

[14] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102, 2018.

[15] Shang Gao, Zecheng Li, Zhe Peng, and Bin Xiao. Power adjusting and bribery racing: Novel mining attacks in the Bitcoin system. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 833–850, 2019.

[16] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin backbone protocol with chains of variable difficulty. In *Annual International Cryptology Conference*, pages 291–323. Springer, 2017.

[17] Juan A Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin backbone protocol: Analysis and applications. *Journal of the ACM*, 2015.

[18] Guy Goren and Alexander Spiegelman. Mind the mining. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 475–487, 2019.

[19] Cyril Grunspan and Ricardo Pérez-Marco. On profitability of selfish mining. *arXiv preprint arXiv:1805.08281*, 2018.

[20] Cyril Grunspan and Ricardo Pérez-Marco. Profit lag and alternate network mining. In *The International Conference on Mathematical Research for Blockchain Economy*, pages 115–132. Springer, 2023.

[21] Ethan Heilman. One weird trick to stop selfish miners: Fresh Bitcoins, a solution for the honest miner. In *International Conference on Financial Cryptography and Data Security*, pages 161–162. Springer, 2014.

[22] Dragos I Ilie, Sam M Werner, Iain D Stewart, and William J Knottenbelt. Unstable throughput: When the difficulty algorithm breaks. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–5. IEEE, 2021.

[23] Aljosha Judmayer, Nicholas Stifter, Philipp Schindler, and Edgar Weippl. Pitchforks in cryptocurrencies: Enforcing rule changes through offensive forking-and consensus techniques (short paper). In *Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2018 International Workshops, DPM 2018 and CBT 2018, Barcelona, Spain, September 6-7, 2018, Proceedings 13*, pages 197–206. Springer, 2018.

[24] Aljosha Judmayer, Nicholas Stifter, Alexei Zamyatin, Itay Tsabary, Ittay Eyal, Peter Gazi, Sarah Meiklejohn, and Edgar Weippl. Pay to win: Cheap, crowdfundable, cross-chain algorithmic incentive manipulation attacks on PoW cryptocurrencies. *Cryptology ePrint Archive*, 2019.

[25] Aljosha Judmayer, Nicholas Stifter, Alexei Zamyatin, Itay Tsabary, Ittay Eyal, Peter Gaži, Sarah Meiklejohn, and Edgar Weippl. Sok: Algorithmic incentive manipulation attacks on permissionless PoW cryptocurrencies. In *Financial Cryptography and Data Security. FC 2021 International Workshops: CoDecFin, DeFi, VOTING, and WTSC, Virtual Event, March 5, 2021, Revised Selected Papers 25*, pages 507–532. Springer, 2021.

[26] Yujin Kwon, Hyoungshick Kim, Jinwoo Shin, and Yongdae Kim. Bitcoin vs. Bitcoin Cash: Coexistence or downfall of Bitcoin Cash? In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 935–951. IEEE, 2019.

[27] Jihye Lee and Yoonjeong Kim. Preventing Bitcoin selfish mining using transaction creation time. In *2018 International Conference on Software Security and Assurance (ICSSA)*, pages 19–24. IEEE, 2018.

[28] Kevin Liao and Jonathan Katz. Incentivizing blockchain forks via whale transactions. In *Financial Cryptography and Data Security: FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers 21*, pages 264–279. Springer, 2017.

[29] Nadisha Madhushanie, Sugandima Vidanagamachchi, and Nalin Arachchilage. Selfish mining attack in blockchain: a systematic literature review. *International Journal of Information Security*, pages 1–19, 2024.

[30] Patrick McCorry, Alexander Hicks, and Sarah Meiklejohn. Smart contracts for bribing miners. In *Financial Cryptography and Data Security: FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort,*

Curaçao, March 2, 2018, Revised Selected Papers 22, pages 3–18. Springer, 2019.

[31] Michael Mirkin, Yan Ji, Jonathan Pang, Ariah Klages-Mundt, Ittay Eyal, and Ari Juels. BDoS: Blockchain denial-of-service. In *Proceedings of the 2020 ACM SIGSAC conference on Computer and Communications Security*, pages 601–619, 2020.

[32] Tejaswi Nadahalli, Majid Khabbazian, and Roger Wattenhofer. Timelocked bribing. In *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part I 25*, pages 53–72. Springer, 2021.

[33] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[34] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 305–320. IEEE, 2016.

[35] Kervins Nicolas, Yi Wang, and George C Giakos. Comprehensive overview of selfish mining and double spending attack countermeasures. In *2019 IEEE 40th Sarnoff Symposium*, pages 1–6. IEEE, 2019.

[36] Saha Reno and Simu Sultana. Preventing selfish mining in public blockchain using alarming block and block interval time approach. In *2022 International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*, pages 988–993. IEEE, 2022.

[37] Meni Rosenfeld. Analysis of Bitcoin pooled mining reward systems. *arXiv preprint arXiv:1112.4980*, 2011.

[38] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in Bitcoin. In *Financial Cryptography and Data Security: 20th International Conference, FC 2016, Christ Church, Barbados, February 22–26, 2016, Revised Selected Papers 20*, pages 515–532. Springer, 2017.

[39] Roozbeh Sarenche, Alireza Aghabagherloo, Svetla Nikova, and Bart Preneel. Bitcoin under volatile block rewards: How mempool statistics can influence Bitcoin mining. *arXiv preprint arXiv:2411.11702*, 2024.

[40] Roozbeh Sarenche, Svetla Nikova, and Bart Preneel. Deep selfish proposing in longest-chain proof-of-stake protocols. *Accepted at Financial Cryptography and Data Security 2024*, 2024. Available at https://eprint.iacr.org/2024/622.

[41] Roozbeh Sarenche, Ren Zhang, Svetla Nikova, and Bart Preneel. Selfish mining time-averaged analysis in Bitcoin: Is orphan reporting an effective countermeasure? *IEEE Transactions on Information Forensics and Security*, 20:449–464, 2025.

[42] Siamak Solat and Maria Potop-Butucaru. Brief announcement: Zeroblock: Timestamp-free prevention of block-withholding attack in Bitcoin. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 356–360. Springer, 2017.

[43] Jason Teutsch, Sanjay Jain, and Prateek Saxena. When cryptocurrencies mine their own business. In *International conference on financial cryptography and data security*, pages 499–514. Springer, 2016.

[44] Itay Tsabary, Matan Yechieli, Alex Manuskin, and Ittay Eyal. MAD-HTLC: because HTLC is crazy-cheap to attack. In *2021 IEEE symposium on security and privacy (SP)*, pages 1230–1248. IEEE, 2021.

[45] Yaron Velner, Jason Teutsch, and Loi Luu. Smart contracts make Bitcoin mining pools vulnerable. In *Financial Cryptography and Data Security: FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers 21*, pages 298–316. Springer, 2017.

[46] Ren Zhang and Bart Preneel. Publish or perish: A backward-compatible defense against selfish mining in Bitcoin. In *Cryptographers' Track at the RSA Conference*, pages 277–292. Springer, 2017.

[47] Ren Zhang and Bart Preneel. Lay down the common metrics: Evaluating proof-of-work consensus protocols' security. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 175–192. IEEE, 2019.

[48] Shiquan Zhang, Kaiwen Zhang, and Bettina Kemme. Analysing the benefit of selfish mining with multiple players. In *2020 IEEE International Conference on Blockchain (Blockchain)*, pages 36–44. IEEE, 2020.

[49] Chencheng Zhou, Liudong Xing, Qisi Liu, and Honggang Wang. Effective selfish mining defense strategies to improve Bitcoin dependability. *Applied Sciences*, 13(1):422, 2022.

[50] Roi Bar Zur, Ittay Eyal, and Aviv Tamar. Efficient MDP analysis for selfish-mining in blockchains. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 113–131, 2020.

Appendix

7.A1 Parameters Involved in Semi-Rational Fork Choice

For $\epsilon = \infty$, the (ϵ, D)-semi-rational fork choice rule is the same as the honest fork choice rule. The ϵ parameter in the definition of the semi-rational fork choice rule captures the idea that a mining pool may decide to stick to the honest strategy up to facing a normalized loss of ϵ. However, once the normalized loss it incurs by following the honest strategy exceeds ϵ, it may deviate from the honest strategy. The parameter D in the definition of the semi-rational fork choice rule represents the maximum length by which the selected chain can be shorter than the longest chain. We use D to present our theorems and lemmas throughout the text. The larger the parameter D, the closer we get to modeling a real-world rational mining pool. However, as D increases, modeling the system becomes more complex, since the number of actions a petty-compliant mining pool might consider also increases.

7.A2 Our System Model Limitations

In our system model, we assume that all block rewards are fixed, and the adversary can place bribes on some blocks to increase the expected return from mining on those blocks. However, this model does not accurately reflect the real world, where blocks can collect varying levels of transaction fees. In Bitcoin, block rewards come from two sources: the protocol reward and the transaction fee reward. In the early years of Bitcoin's introduction, the protocol reward was much higher than the transaction fees, making the fixed block reward assumption reasonable. However, as Bitcoin undergoes more halving events, the protocol reward decreases, and transaction fees become the primary source of mining rewards. In the new transaction-fee era, different blocks can have varying rewards, even without bribes being placed on top of them. This can significantly impact mining power destruction attacks. For instance, in a block race, petty-compliant mining pools will typically adopt the block that leaves the higher transaction fee in the mempool, which is usually the block mined earlier. Therefore, a selfish miner who mines the first block and keeps it secret has a higher chance of winning the block race compared to the miner of the other

block in the public fork, suggesting that these attacks may be more threatening in the transaction-fee era. Several papers in the literature [8, 9, 11, 39] have analyzed Bitcoin mining destruction attacks in the presence of altruistic or infinitesimally petty-compliant nodes. However, studying mining destruction attacks in the presence of petty-compliant mining pools in the transaction-fee era remains an interesting direction for future research.

Another assumption in our model is that there is a single adversarial mining pool in the system, while the remaining pools are petty-compliant. This limits the strategies of the remaining petty-compliant pools to a predefined set of actions, implying they cannot respond arbitrarily to the attack. However, in practice, multiple mining pools may behave adversarially to maximize their payoffs. For example, once an adversarial mining pool offers a bribe to launch an attack, another pool may respond by offering a higher bribe to defend against the attack. Several papers in the literature have analyzed multi-agent selfish mining attacks [7, 48], although they do not consider bribes and transaction fees. Analyzing multi-agent destruction attacks in the presence of bribes and within the transaction-fee era could be another interesting direction for future research.

7.A3 Selfish Mining Attack: Supplementary

7.A3.1 Whale Transactions Used as Bribes

Once the adversarial mining pool mines its first block in its private fork, it publishes transaction tx_1 in the public transaction pool. If the transaction fee of tx_1 is chosen properly, tx_1 would be included in the first block of the public non-adversarial fork. The adversarial mining pool refrains from including tx_1 in its private fork. Once the non-adversarial fork length becomes equal to the adversarial fork, the adversarial mining pool publishes its private fork along with transaction tx_2 that is only valid if tx_1 is not already included in the chain (both tx_1 and tx_2 pays out form the same address). The transaction fee of tx_2 can be set to ϵR. In this case, the indifferent petty-compliant mining pools get incentivized to select the adversarial fork since they can include tx_2 only in the block mined on top of the adversarial fork, not the non-adversarial fork.

7.A3.2 Proof of Lemma 7.1

Proof. Let $\mathcal{P}_{\overline{\mathcal{A}}}$ denote the set of all mining pools excluding the adversarial mining pool. Assume that the single block available in the non-adversarial fork is mined by petty-compliant mining pool $p_i \in \mathcal{P}_{\overline{\mathcal{A}}}$, whose mining share is equal

to α_i. The probability of this event is equal to α_i. Since there is a bribe of ϵR available on top of the adversarial fork, all the mining pools excluding p_i select the adversarial fork to mine on top of. This implies the only mining pool that is incentivized to mine on top of the non-adversarial fork is the mining pool p_i itself. Therefore, given that the first block of the non-adversarial fork is mined by the petty-compliant mining pool p_i, the probability that the next block is mined on top of the non-adversarial fork is equal to α_i. Consequently, the total probability of the event that the next block is mined on top of the non-adversarial fork (denoted by event E) can be obtained as follows:

$$\texttt{Pr}(E) = \frac{\sum_{p_i \in \mathcal{P}_{\overline{\mathcal{A}}}} {\alpha_i}^2}{\sum_{p_i \in \mathcal{P}_{\overline{\mathcal{A}}}} \alpha_i} = \frac{\sum_{p_i \in \mathcal{P}_{\overline{\mathcal{A}}}} {\alpha_i}^2}{1 - \alpha_{\mathcal{A}}} = \beta_{\mathcal{A}}, \tag{7.6}$$

where $\alpha_{\mathcal{A}}$ and $\beta_{\mathcal{A}}$ represent the mining share and the residual centralization factor of the adversarial mining pool, respectively. The event mentioned in the lemma is complementary to event E, and thus, its probability is equal to $1 - \beta_{\mathcal{A}}$. □

7.A3.3 Selfish Mining Strategy

In this section, we introduce the selfish mining strategy π^{selfish}, which is suitable for an $(\epsilon, D = 1)$-semi-rational environment, where the mining share of all mining pools excluding the adversarial mining pool is assumed to be less than 0.4302. This assumption is important as it helps us analyze the strategy of a petty-compliant pool that is under the selfish mining π^{selfish} attack.

Lemma 7.2. Assume mining pool p_i to be an $(\epsilon, D = 1)$-petty-compliant mining pool. Consider a fork race between forks f_1 and f_2, each of length $l_1 = 1$ and $l_2 = 2$, respectively, where the single block on fork f_1 is mined by mining pool p_i. The mining pool p_i is incentivized to abandon mining on fork f_1 to mine on top of f_2 if $\alpha_i < 0.4302$.

The proof of Lemma 7.2 is presented in App. 7.A3.4. Note that in the altruistic setting, we can always assume that once a mining pool finds its fork shorter than the adversarial fork, it will adopt the adversarial fork. However, this behavior is not necessarily followed by a petty-compliant mining pool under the attack. Lemma 7.2 shows that, if under a selfish mining attack, an $(\epsilon, D = 1)$-petty-compliant mining pool with mining share less than 0.4302 finds its one-block fork shorter than the adversarial fork, it is also incentivized to abandon its fork and mine on top of the adversarial fork.

Definition 7.A3.1 (Strategy π^{selfish})**.** Let $l_{\mathcal{A}}$ and $l_{\mathcal{R}}$ denote the length of the adversarial and semi-rational forks, respectively. Strategy π^{selfish} is defined as follows:

- Assume the initial fork state is $(l_{\mathcal{A}}, l_{\mathcal{R}}) = (0, 0)$. If a petty-compliant mining pool mines a new block, the adversarial mining pool adds the semi-rational block to the head of its chain and continues mining on top of that. Therefore, the fork state remains in state $(l_{\mathcal{A}}, l_{\mathcal{R}}) = (0, 0)$.
- Assume the initial fork state is $(l_{\mathcal{A}}, l_{\mathcal{R}}) = (0, 0)$. If the adversarial mining pool mines a new block, it keeps the block hidden and does not publish it to the petty-compliant pools. Therefore, the fork state transitions to state $(l_{\mathcal{A}}, l_{\mathcal{R}}) = (1, 0)$.
- Assume the initial fork state is $(l_{\mathcal{A}}, l_{\mathcal{R}}) = (1, 0)$. If a petty-compliant mining pool p_i mines a new block, a fork race occurs between the adversarial and the semi-rational blocks, and the fork state transitions to state $(l_{\mathcal{A}}, l_{\mathcal{R}}) = (1, 1)$. In this case, the adversarial mining pool publishes its block along with a normalized bribe of $\mathtt{br} = \epsilon$ on top of it. As a result of this bribe, all petty-compliant pools, except p_i, mine on top of the adversarial fork, while mining pool p_i continues to mine on top of its own block.
- Assume the initial fork state is $(l_{\mathcal{A}}, l_{\mathcal{R}}) = (1, 0)$. If the adversarial mining pool mines a new block, the fork state transitions to state $(l_{\mathcal{A}}, l_{\mathcal{R}}) = (2, 0)$.
- Assume the initial fork state is $(l_{\mathcal{A}}, l_{\mathcal{R}}) = (1, 1)$. If the next block is mined on top of the semi-rational fork, the adversarial mining pool gives up on its block and admits the semi-rational fork as the canonical fork. Therefore, the fork state transitions to state $(l_{\mathcal{A}}, l_{\mathcal{R}}) = (0, 0)$. In this case, the adversarial mining pool retracts its bribe and avoids incurring the bribe cost.
- Assume the initial fork state is $(l_{\mathcal{A}}, l_{\mathcal{R}}) = (1, 1)$. If the next block is mined on top of the adversarial fork, the length of the adversarial fork exceeds that of the semi-rational fork by one block. In this case, we argue that all mining pools adopt the longer adversarial fork and mine on top of it. For a petty-compliant pool p_j that is not the miner of the single block in the semi-rational fork ($p_j \neq p_i$), it is evident that it adopts the longer adversarial fork, as there is no incentive to mine on top of the shorter semi-rational fork owned by another pool. According to Lemma 7.2 and the assumption $\alpha_i < 0.4302$, even the petty-compliant mining pool p_i, which mined the single semi-rational block, is incentivized to abandon its block and adopt the adversarial fork as the canonical fork. Therefore, the fork state transitions to state $(l_{\mathcal{A}}, l_{\mathcal{R}}) = (0, 0)$. Note that, in this case, the bribe becomes payable only if the adversarial fork is extended by a non-adversarial block.
- Assume the initial fork state is $(l_{\mathcal{A}}, l_{\mathcal{R}}) = (2, 0)$. If a petty-compliant mining pool p_i mines a new block, the adversarial mining pool publishes its 2 hidden blocks. Similar to the previous case, according to Lemma 7.2 and under the assumption of $\alpha_i < 0.4302$, the petty-compliant mining pool

π gives up on its block and admits the adversarial fork as the canonical fork. Therefore, the fork state transitions to state $(l_{\mathcal{A}}, l_{\mathcal{R}}) = (0, 0)$.

- Assume the initial fork state is $(l_{\mathcal{A}}, l_{\mathcal{R}}) = (2, 0)$. If the adversarial mining pool mines a new block, the adversarial mining pool publishes its first hidden block and remains in the state $(l_{\mathcal{A}}, l_{\mathcal{A}}) = (2, 0)$. [5]

7.A3.4 Proof of Lemma 7.2

Before proving Lemma 7.2, we first review some useful lemmas. To calculate the expected return of different forks in the view of a given petty-compliant mining pool p, we use random walking on a (x, y)-grid. The two-dimensional (x, y)-grid represents the fork race between the fork of mining pool p and the fork of remaining mining pools, referred to as the public fork. Each fork race scenario can be represented by a path on this grid, which is referred to as the *mining path*. Whenever the petty-compliant mining pool p mines a new block, the mining path moves one step to the right, and whenever a block is added on top of the public fork, the mining path moves one step up.

Lemma 7.3. Consider the (x, y)-grid representation of the fork race between the petty-compliant mining pool p with mining share α and the remaining mining pools. Let $\overline{P_r}$ for $r \geq 1$ denote the probability of the event that the mining path starting at $(0, 0)$ never reaches the line $y = x - r$. We have $\overline{P_r} = 1 - \left(\frac{\alpha}{1-\alpha}\right)^r$.

Proof. The proof of Lemma 7.3 is presented in App. D.2 of [40]. Here, we briefly review the idea behind the proof.

Let P_r denote the probability of the event that the mining path starting at $(0, 0)$ reaches the line $y = x - r$ at least once. Starting at point $(0, 0)$, the mining path moves to point $(1, 0)$ with probability α and moves to point $(0, 1)$ with probability $1 - \alpha$. The probability of the event that the mining path starting at $(1, 0)$ reaches the line $y = x - r$ is equal to P_{r-1}. And, the probability of the event that the mining path starting at $(0, 1)$ reaches the line $y = x - r$ is equal to P_{r+1}. Therefore, we have: $P_r = \alpha P_{r-1} + (1 - \alpha)P_{r+1}$. This recursive equation can be evaluated for different values of $r \geq 1$ by having P_1 and P_2.

We first find the probability of P_1, i.e., the probability of reaching the line $y = x - 1$ starting at point $(0, 0)$. Suppose the mining path reaches the line $y = x - 1$ at the point $(s + 1, s)$ for the first time. For this to occur, the mining path must reach the point (s, s) without crossing below the line $y = x$, and

[5] This action simplifies the relative revenue calculation and is not the optimal action that the adversarial mining pool can take in state $(l_{\mathcal{A}}, l_{\mathcal{R}}) = (2, 0)$. In an optimal strategy, the adversarial mining pool may hide all three blocks to orphan a longer non-adversarial fork.

then take a step to the right. The number of valid paths from $(0,0)$ to (s,s), while not passing below the line $y = x$, corresponds to the s-th Catalan number, which we refer to as c_s. Therefore, the probability that the mining path reaches the line $y = x - 1$ for the first time at $(s+1, s)$ is given by $c_s \alpha^{s+1}(1-\alpha)^s$. Hence, the overall probability of reaching the line $y = x - 1$ can be expressed as:

$$P_1 = \alpha \sum_{s=0}^{\infty} c_s \alpha^s (1-\alpha)^s = \frac{\alpha}{1-\alpha} \ . \tag{7.7}$$

The method for solving this series is provided in [12]. We next find the probability of P_2, i.e., the probability of reaching the line $y = x - 1$ starting at point $(0,0)$. Suppose the mining path first reaches the line $y = x - 2$ at the point $(s+2, s)$. For this to happen, the mining path must first reach the point $(s+1, s)$ without falling below the line $y = x-1$, and then take one step to the right. The number of valid paths from $(0,0)$ to $(s+1, s)$, without falling below the line $y = x - 1$, corresponds to the $(s+1)$-th Catalan number, c_{s+1}. Thus, the probability of reaching the line $y = x - 2$ for the first time at the point $(s+2, s)$ is given by $c_{s+1}\alpha^{s+2}(1-\alpha)^s$. Therefore, the probability of reaching the line $y = x - 2$ is:

$$P_2 = \alpha^2 \sum_{s=0}^{\infty} c_{s+1} \alpha^s (1-\alpha)^s = \frac{\alpha}{1-\alpha} \sum_{s=1}^{\infty} c_s \alpha^s (1-\alpha)^s = \left(\frac{\alpha}{1-\alpha}\right)^2 \ . \tag{7.8}$$

Knowing $P_1 = \frac{\alpha}{1-\alpha}$, $P_2 = \left(\frac{\alpha}{1-\alpha}\right)^2$, and the recursive equation $P_r = \alpha P_{r-1} + (1-\alpha)P_{r+1}$, the probability P_r for $r \geq 1$ can be obtained as $P_r = \left(\frac{\alpha}{1-\alpha}\right)^r$. Therefore, the probability of the event that the mining path starting at $(0,0)$ never reaches the line $y = x - r$ can be obtained as $\overline{P_r} = 1 - \left(\frac{\alpha}{1-\alpha}\right)^r$.

□

Lemma 7.4. Let G_s^d for $d \geq 1$ denote the number of paths in a (x,y)-grid from start point $(0,0)$ to the point $(s, s+d)$ without reaching the lines $y = x - 2$

and $y = x + d$ in advance. We have:

$$\sum_{s=0}^{\infty} G_s^d \big(\alpha(1-\alpha)\big)^s = \frac{1-2\alpha}{(1-\alpha)^{d+2} - \alpha^{d+2}} \ ,$$

$$\sum_{s=0}^{\infty} s G_s^d \big(\alpha(1-\alpha)\big)^s = \frac{(d+2)\alpha(1-\alpha)\big((1-\alpha)^{d+1} + \alpha^{d+1}\big)}{\big((1-\alpha)^{d+2} - \alpha^{d+2}\big)^2} \tag{7.9}$$

$$- \frac{2\alpha(1-\alpha)}{(1-2\alpha)\big((1-\alpha)^{d+2} - \alpha^{d+2}\big)} \ .$$

Proof. The proof follows the same logic as the proof of Lemma 4 presented in App. D.2 of [40]. Assume $\texttt{Event}_1(s)$ is defined as follows: (i) the block path starting at $(0,0)$ reaches the point $(s, s+d-1)$, and (ii) before reaching the point $(s, s+d-1)$, the block path never passes the line $y = x+d-1$ and never reaches the line $y = x-2$. According to the definition of G_s^d, the probability of $\texttt{Event}_1(s)$ is equal to $G_s^d \alpha^s (1-\alpha)^{s+d-1}$. Assume $\texttt{Event}_2(s)$ is defined as follows: (i) the block path starting at $(0,0)$ passes the line $y = x+d-1$ for the first time at point $(s, s+d-1)$, and (ii) before reaching the point $(s, s+d-1)$, the block path never reaches the line $y = x-2$. $\texttt{Event}_2(s)$ happens if, after the occurrence of $\texttt{Event}_1(s)$ and reaching the point $(s, s+d-1)$, the block path immediately moves one step up to pass the line $y = x+d-1$ and reach the point $(s, s+d)$. Therefore, the probability of $\texttt{Event}_2(s)$ is equal to $G_s^d \alpha^s (1-\alpha)^{s+d}$. Assume $\texttt{Event}_3(s)$ is defined as follows: (i) the block path starting at $(0,0)$ reaches the line $y = x+d$ for the first time at point $(s, s+d)$, and (ii) the block path never reaches the line $y = x-2$ both before and after reaching the point $(s, s+d)$. $\texttt{Event}_3(s)$ happens if, after the occurrence of $\texttt{Event}_2(s)$ and reaching the point $(s, s+d)$, the block path never reaches the line $y = x-r$. The event that the block path starting at $(s, s+d)$ never reaches the line $y = x-2$ is equivalent to the event that the block path starting at $(0,0)$ never reaches the line $y = x-(d+2)$, the probability of which is equal to $1 - (\frac{\alpha}{1-\alpha})^{d+2}$ according to Lemma 7.3. Therefore, the probability of $\texttt{Event}_3(s)$ is equal to $G_s^d \alpha^s (1-\alpha)^{s+d} \big(1 - (\frac{\alpha}{1-\alpha})^{d+2}\big)$. Note that since $1-\alpha > \alpha$, a mining path will eventually pass the line $y = x+d$. Therefore, the sum of $\texttt{Event}_3(s)$ probabilities over all values of s is equal to the probability that the block path never reaches the line $y = x-2$, the probability of which is equal to $1 - (\frac{\alpha}{1-\alpha})^2$ according to

Lemma 7.3. As a result,

$$\sum_{s=0}^{\infty} G_s^d \alpha^s (1-\alpha)^{s+d} \big(1 - (\frac{\alpha}{1-\alpha})^{d+2}\big) = 1 - (\frac{\alpha}{1-\alpha})^2$$
$$\Rightarrow \sum_{s=0}^{\infty} G_s^d \big(\alpha(1-\alpha)\big)^s = \frac{1-2\alpha}{(1-\alpha)^{d+2} - \alpha^{d+2}} \ . \tag{7.10}$$

To prove the second equality, we use the variable substitution $\alpha(1-\alpha) = x$ in the equality above. By taking the derivative from both sides, then multiplying both sides to x, and finally substituting $x = \alpha(1-\alpha)$, the second equality can be obtained. □

Lemma 7.5. Let F_s^d for $d \geq 1$ denote the number of paths in a (x, y)-grid from start point $(0, 0)$ to the point $(s+2, s)$ without reaching the lines $y = x - 2$ and $y = x + d$ in advance. We have:

$$\sum_{s=0}^{\infty} F_s^d \big(\alpha(1-\alpha)\big)^s = \frac{1}{\alpha^2}\Big(1 - \frac{1-2\alpha}{(1-\alpha)^2\big(1-(\frac{\alpha}{1-\alpha})^{d+2}\big)}\Big) \ . \tag{7.11}$$

Proof. According to the proof of Lemma 7.4, the probability of the event that the mining path starting at point $(0, 0)$ reaches the line $y = x+d$ before reaching the line $y = x - 2$ is equal to $\frac{1-2\alpha}{(1-\alpha)^2\left(1-(\frac{\alpha}{1-\alpha})^{d+2}\right)}$. Therefore, the probability of the event that the mining path starting at point $(0, 0)$ reaches the line $y = x - 2$ before reaching the line $y = x + d$ is equal to $1 - \frac{1-2\alpha}{(1-\alpha)^2\left(1-(\frac{\alpha}{1-\alpha})^{d+2}\right)}$. According o the definition of F_s^d, the latter probability is equal to $F_s^d \alpha^2 \big(\alpha(1-\alpha)\big)^s$. This completes our proof. □

Proof of Lemma 7.2. We need to compare the expected return of mining on top of 2 forks f_1 and f_2 to find out which one is more profitable. We assume:

- If fork f_1 lags behind fork f_2 by d blocks, where $d \geq 2$, the mining pool p_i switches to mine on top of f_2. This is because mining pool p_i is assumed to be a $D = 1$-petty-compliant mining pool.
- If fork f_1 becomes even one block longer than fork f_2, all the remaining mining pools switch to mine on top of f_1. This assumption is in favor of the mining pool p_i.

We calculate the expected return using the (x, y)-grid. $l_1 = 1$ and $l_2 = 2$ implies that the mining path is at point $(1, 2)$. If the mining path reaches the line $y = x - 1$ before reaching the line $y = x + d$, the mining pool p_i wins the fork race. If the mining path reaches the line $y = x + d$ before reaching the line $y = x - 1$, the mining pool p_i loses the fork race. Assume the mining path

starting at point $(1, 2)$ reaches the line $y = x - 1$ for the first time at point $(s + 3, s + 2)$ without reaching the line $y = x + d$ in advance. The probability of this event is equal to $F_s^{d-1}\alpha_i^2(\alpha_i(1-\alpha_i))^s$. The mining pool p_i receives $s + 3$ block rewards under this event. Therefore, the expected return of mining on top of fork f_1 for mining pool p_i up to the point that the fork race is resolved can be obtained as follows:

$$r_1 = \sum_{s=0}^{\infty} (s+3)F_s^{d-1}\alpha_i^2(\alpha_i(1-\alpha_i))^s \quad . \tag{7.12}$$

The fork race ends once the mining path reaches either $y = x - 1$ or $y = x + d$. To obtain the expected return of switching the fork, we calculate the reward that the mining pool p_i could have received if it was mining on top of the fork f_2. Assume the mining path starting at point $(1, 2)$ reaches the line $y = x + d$ for the first time at point $(s+1, s+d+1)$ without reaching the line $y = x - 1$ in advance. The probability of this event is equal to $G_s^{d-1}(1-\alpha_i)^{d-1}(\alpha_i(1-\alpha_i))^s$. The mining pool p_i receives s block rewards under this event. Now, assume the mining path starting at point $(1, 2)$ reaches the line $y = x - 1$ for the first time at point $(s + 3, s + 2)$ without reaching the line $y = x + d$ in advance. The probability of this event is equal to $F_s^{d-1}\alpha_i^2(\alpha_i(1-\alpha_i))^s$. The mining pool p_i receives $s + 2$ block rewards under this event. Therefore, the expected return of mining on top of fork f_2 for mining pool p_i up to the point that the mining path reaches either $y = x - 1$ or $y = x + d$ can be obtained as follows:

$$r_2 = \sum_{s=0}^{\infty} (s+2)F_s^{d-1}\alpha_i^2(\alpha_i(1-\alpha_i))^s + sG_s^{d-1}(1-\alpha_i)^{d-1}(\alpha_i(1-\alpha_i))^s \quad . \tag{7.13}$$

Using Lemmas 7.5 and 7.4, r_1 and r_2 can be calculated. The petty-compliant mining pool p_i is incentivized to leave its fork at state $l_1 = 1$ and $l_2 = 2$ if $r_1 - r_2 < 0$. For $\alpha_i \in (0, 0.5)$, the function $r_1 - r_2$ is decreasing on d. By assigning $d = 2$, the inequality $r_1 - r_2 < 0$ holds if $\alpha_i < 0.4302$. □

7.A3.5 Proof of Thm. 7.1

Proof of Thm. 7.1. Let $\mathtt{Pr}_{l_{\mathcal{A}},l_{\mathcal{R}}}$ denote the probability of being in state $(l_{\mathcal{A}}, l_{\mathcal{R}})$. We can obtain the following equations:

$$\begin{aligned}
\mathtt{Pr}_{0,0} &= \frac{1-\alpha_{\mathcal{A}}}{1+(1-\alpha_{\mathcal{A}})^2\alpha_{\mathcal{A}}} \ , \\
\mathtt{Pr}_{1,0} &= \alpha_{\mathcal{A}}\mathtt{Pr}_{0,0} \ , \\
\mathtt{Pr}_{2,0} &= \frac{\alpha_{\mathcal{A}}}{1-\alpha_{\mathcal{A}}}\mathtt{Pr}_{1,0} \ , \\
\mathtt{Pr}_{1,1} &= (1-\alpha_{\mathcal{A}})\mathtt{Pr}_{1,0} \ .
\end{aligned} \tag{7.14}$$

To find the adversarial average revenue, we should consider the average number of adversarial blocks that get added to the canonical chain at each state. The average revenue that the adversarial mining pool receives can be obtained as follows:

$$\textsc{Rev} = \Big(\alpha_{\mathcal{A}}\mathtt{Pr}_{2,0}+2(1-\alpha_{\mathcal{A}})\mathtt{Pr}_{2,0}+2\alpha_{\mathcal{A}}\mathtt{Pr}_{1,1}+(1-\alpha_{\mathcal{A}}-\beta_{\mathcal{A}})\mathtt{Pr}_{1,1}\Big)\lambda' R \ , \tag{7.15}$$

where λ' is the block mining rate under the selfish mining attack.

To find the adversary's cost, we ignore the mining cost[6] and only consider the cost incurred by the adversarial mining pool due to paying bribes. Note that the normalized bribe $\mathtt{br} = \epsilon$ is payable if, under a fork race, the adversarial fork is extended by a non-adversarial block. In this case, the non-adversarial mining pool that extends the adversarial block is eligible to collect the bribe. The average cost of the adversarial mining pool can be obtained as follows:

$$\textsc{Cost} = \sum_{p_i \in \mathcal{P}_{\overline{\mathcal{A}}}} \epsilon(1-\alpha_{\mathcal{A}}-\alpha_i)\alpha_i \mathtt{Pr}_{1,0}\lambda' R = \epsilon(1-\alpha_{\mathcal{A}}-\beta_{\mathcal{A}})\mathtt{Pr}_{1,1}\lambda' R \ , \tag{7.16}$$

where $\mathcal{P}_{\overline{\mathcal{A}}}$ is the set of non-adversarial mining pools.

The mining rate under the selfish mining attack can be obtained as $\lambda' = \frac{\lambda}{\mathtt{Diff}}$, where $\mathtt{Diff}$ is the normalized number of blocks added to the canonical chain and can be obtained as follows:

$$\mathtt{Diff} = (1-\alpha_{\mathcal{A}})\mathtt{Pr}_{0,0} + \alpha_{\mathcal{A}}\mathtt{Pr}_{2,0} + 2(1-\alpha_{\mathcal{A}})\mathtt{Pr}_{2,0} + 2\mathtt{Pr}_{1,1} \ . \tag{7.17}$$

[6] The mining cost during both honest mining and selfish mining is the same, and thus does not affect the profit comparison between these two strategies.

Table 7.1: Mining Power Distribution of Bitcoin Mining Pools

Foundry USA	**AntPool**	**ViaBTC**	**F2Pool**	**Unknown**	**Mara Pool**
29.03%	24.85%	12.86%	11.53%	7.90%	3.44%
Binance Pool	**SBI Crypto**	**Braiins Pool**	**BTC.com**	**BTC M4**	**Poolin**
3.00%	2.10%	1.78%	1.42%	0.83%	0.80%
Ultimus	**1THash**	**Solo CKPool**	**KanoPool**		
0.35%	0.00054%	0.034%	0.011%		

Therefore, after mining difficulty adjustment, the time-averaged profit can be calculated as below:

$$\textsc{Profit}\big(\pi^{\text{selfish}}\big) = \frac{2\alpha_{\mathcal{A}}^4 - 5\alpha_{\mathcal{A}}^3 + 4\alpha_{\mathcal{A}}^2 + \alpha_{\mathcal{A}}(1-\alpha_{\mathcal{A}})(1-\alpha_{\mathcal{A}}-\beta_{\mathcal{A}})(1-\epsilon)}{\alpha_{\mathcal{A}}^3 - \alpha_{\mathcal{A}}^2 + 1}\lambda R \ . \tag{7.18}$$

The selfish mining strategy π^{selfish} strongly dominates honest mining if we have:

$$\textsc{Profit}\big(\pi^{\text{selfish}}\big) > \alpha_{\mathcal{A}}\lambda R \iff \beta_{\mathcal{A}} < \frac{\alpha_{\mathcal{A}} - \epsilon(1-\alpha_{\mathcal{A}})^2}{(1-\alpha_{\mathcal{A}})(1-\epsilon)} \ . \tag{7.19}$$

□

7.A3.6 Proof of Cor. 7.1

Proof. According to Thm. 7.1, selfish mining dominates honest mining for P_1 if the inequality $\beta_1 < \frac{\alpha_1 - \epsilon(1-\alpha_1)^2}{(1-\alpha_1)(1-\epsilon)}$ holds. The maximum value that P_1's residual centralization factor β_1 can take is equal to the mining share of the largest mining pool excluding P_1, which in our case is α_2. This maximum value occurs when all mining pools, excluding P_1, have the same mining share α_i. Therefore, to satisfy the inequality $\beta_1 < \frac{\alpha_1 - \epsilon(1-\alpha_1)^2}{(1-\alpha_1)(1-\epsilon)}$, one must ensure that $\alpha_2 < \frac{\alpha_1 - \epsilon(1-\alpha_1)^2}{(1-\alpha_1)(1-\epsilon)}$ holds. The latter inequality holds if $\frac{\alpha_1}{1-\alpha_1} > \alpha_2 + \epsilon(1-\alpha_1-\alpha_2)$ holds. This proves the first statement of the corollary.

In the case of $\epsilon = 0$, selfish mining is the dominant strategy for P_1 if $\frac{\alpha_1}{1-\alpha_1} > \alpha_2$ holds, which is always the case as $\alpha_1 \geq \alpha_2$. □

7.A3.7 Mining Pools

The distribution of mining power share for the first 8 months of 2024 is presented in Table 7.1 [3]. We consider all the unknown miners as a single mining pool.

7.A3.8 MDP-Based Analysis

In this section, we present an MDP-based implementation to analyze selfish mining in the presence of petty-compliant mining pools. The implementation is available at [2]. Our MDP implementation considers the following two assumptions: 1) The environment is an $(\epsilon, D = 0)$-semi-rational environment, where $D = 0$ implies that petty-compliant mining pools always select the longest chain. However, in the case of a fork with chains of the same height, they opt for the chain with the highest return. 2) The bribe placed on the adversarial chain is valid until the next non-adversarial block is mined. The next non-adversarial block either collects the bribe or leaves it unspent, allowing the adversary to collect it back. This assumption can be implemented using a smart contract, as described in App. 7.A3.9. Our selfish mining analysis differs from the implementation in [9], which assumes volatile block rewards. Our approach assumes uniform block rewards, while allowing the adversary to place bribes on top of the adversarial fork.

MDP can be used to obtain the optimal strategy in settings where the number of states is limited to less than 10^7 [47]. Due to the limitation of the state space, our MDP implementation only accounts for sufficiently large mining pools, while treating the remaining mining power as a single, aggregated mining pool. Let N denote the total number of mining pools available in the network. Our implementation can consider up to almost $N = 10$ mining pools. Assume that mining pool p_1 is the single honest mining pool.[7] Assume further that the set $\{p_2, p_3, \ldots, p_{N-1}\}$ denotes the $N - 2$ petty-compliant mining pools, and $p_{\mathcal{A}}$ denotes the adversarial mining pool. Each state in our implementation represents a fork race between the adversarial fork of length $l_{\mathcal{A}}$ and the non-adversarial fork of length $l_{\overline{\mathcal{A}}}$. We first discuss the set of possible actions. The adversarial mining pool can choose from four major actions: `Override`, `Wait`, `Adopt`, and `Match`, where the action `Match` consists of multiple subactions.

`Override`: This action represents the adversarial mining pool publishing a sub-fork of its secret fork that is one block longer than the honest fork. As a result of the `Override` action, non-adversarial mining pools abandon their current fork and start mining on top of the longer adversarial fork. This is because, according to our assumption, the petty-compliant pools are $(\epsilon, D = 0)$-semi-rational, meaning that they always mine on top of the longest chain. This action is feasible if $l_{\mathcal{A}} > l_{\overline{\mathcal{A}}}$.

`Wait`: This action represents the adversarial mining pool continuing to mine new blocks on top of its secret fork. This action is always feasible.

[7] All honest mining power is unified into a single honest mining pool.

`Adopt`: The `Adopt` action represents the adversarial mining pool abandoning its adversarial fork and accepting the non-adversarial fork. This action is always feasible.

$\texttt{Match}_i$: The action $\texttt{Match}_i$ for $i \in \{0, 1, \cdots, \texttt{max_bribe}\}$ represents the adversarial mining pool publishing a sub-fork of its secret fork that is equal in length to the non-adversarial fork and placing a normalized bribe of value $i + \epsilon$ on top of the adversarial fork. `max_bribe` is a parameter denoting the maximum bribe the adversary can pay. Action $\texttt{Match}_i$ results in a race between two forks of the same height. Petty-compliant pools, whose number of blocks on the non-adversarial fork is less than or equal to i, are incentivized to mine on the adversarial fork. This action is feasible if $l_{\mathcal{A}} \geq l_{\overline{\mathcal{A}}}$.

The most important difference between the set of possible actions in the setting of petty-compliant mining pools and the altruistic setting is the action $\texttt{Match}_i$ for $i \in [\texttt{max_bribe}]$. Note that the undercutting attack is a subset of the stated possible actions. The undercutting attack can be described as follows: starting in state $(l_{\mathcal{A}} = 0, l_{\overline{\mathcal{A}}} = 1)$, the adversarial mining pool first takes the `Wait` action, attempts to mine a block to transition to state $(l_{\mathcal{A}} = 1, l_{\overline{\mathcal{A}}} = 1)$, and then proceeds with the $\texttt{Match}_0$ action.

Each state in our implementation has the following form:

$$\left(l_1, l_2, \ldots, l_{N-1}, l_{\mathcal{A}}, \texttt{latest}, \texttt{match}, \texttt{bribe}\right) .$$

Here, l_i for $i \in [N-1]$ denotes the number of blocks mined by pool p_i in the non-adversarial fork, where we have $l_{\overline{\mathcal{A}}} = \sum_{i=1}^{N-1} l_i$. The variable `latest` represents information on the latest block mined in the system, `match` is a boolean element indicating whether the action `Match` is active, and `bribe` denotes the amount of bribe placed on top of the adversarial fork. Table 7.2 presents the reward shares that an adversarial mining pool with a mining share of 0.4 can achieve under different distributions of mining power among petty-compliant mining pools. Table 7.3 presents the selfish mining reward shares of an adversarial mining pool with a mining share of 0.3. Table 7.4 presents the selfish mining reward shares achieved by mining pools, with mining shares assigned according to the real-world distribution shown in Table 7.1. As can be observed, in scenarios where the residual centralization factor with respect to the adversary is lower, the reward share the adversary can achieve is higher.

7.A3.9 Smart Contract-Based Selfish Mining

The goal of the adversarial mining pool is to design a bribing smart contract such that the bribe placed on the adversarial fork remains valid until the next non-adversarial block is mined. If a non-adversarial block is mined on top of

Table 7.2: Selfish mining in the presence of petty-compliant pools ($\alpha_{\mathcal{A}} = 0.4,\ \epsilon = 0.1$).

Adversarial Share	**Petty-Compliant Mining Pool Shares**
0.40	[0.3, 0.3]
Residual Centralization Factor: 0.3	**Reward Share:** 0.5448
Adversarial Share	**Petty-Compliant Mining Pool Shares**
0.40	[0.2, 0.2, 0.2]
Residual Centralization Factor: 0.2	**Reward Share:** 0.5714
Adversarial Share	**Petty-Compliant Mining Pool Shares**
0.40	[0.1, 0.1, 0.1, 0.1, 0.05, 0.05, 0.05, 0.05]
Residual Centralization Factor: 0.0766	**Reward Share:** 0.5955
Adversarial Share	**Petty-Compliant Mining Pool Shares**
0.40	[0.075, 0.075, 0.075, 0.075, 0.075, 0.075, 0.075, 0.075]
Residual Centralization Factor: 0.075	**Reward Share:** 0.5967

the adversarial fork, the miner of the non-adversarial block should be able to receive the bribe; otherwise, the adversary should be able to recollect it.

Let $B_{\mathcal{A}}$ denote the tip of the adversarial fork and $B_{\overline{\mathcal{A}}}$ denote the tip of the non-adversarial fork. The adversarial mining pool deploys a smart contract and publishes it for all mining pools. The smart contract stores four parameters: the hash of block $B_{\mathcal{A}}$, denoted by $\texttt{Hash}(B_{\mathcal{A}})$; the hash of block $B_{\overline{\mathcal{A}}}$, denoted by $\texttt{Hash}(B_{\overline{\mathcal{A}}})$; a difficulty target denoted by `Target`, which is equal to the difficulty target of the epoch to which blocks $B_{\mathcal{A}}$ and $B_{\overline{\mathcal{A}}}$ belong; and a payout address $\texttt{add}_{\mathcal{A}}$ for the adversarial mining pool. The adversarial mining pool deposits a normalized bribe `br` into the smart contract. Anyone who submits a witness proving the mining of a block on top of the adversarial block $B_{\mathcal{A}}$ can withdraw `br`. A valid witness includes a Bitcoin `nonce`, a Merkle root `MR`, a Bitcoin coinbase transaction tx, a Merkle inclusion `proof`, and a payout address `add` in the host blockchain, satisfying the following conditions:

Table 7.3: Selfish mining in the presence of petty-compliant pools ($\alpha_{\mathcal{A}} = 0.3,\ \epsilon = 0$).

Adversarial Share	**Petty-Compliant Mining Pool Shares**
0.30	[0.4, 0.2, 0.1]
Residual Centralization Factor: 0.3	**Reward Share:** 0.3534
Adversarial Share	**Petty-Compliant Mining Pool Shares**
0.30	[0.2, 0.2, 0.2, 0.1]
Residual Centralization Factor: 0.1857	**Reward Share:** 0.3877
Adversarial Share	**Petty-Compliant Mining Pool Shares**
0.30	[0.2, 0.2, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05]
Residual Centralization Factor: 0.1357	**Reward Share:** 0.4006
Adversarial Share	**Petty-Compliant Mining Pool Shares**
0.30	[0.0875, 0.0875, 0.0875, 0.0875, 0.0875, 0.0875, 0.0875, 0.0875]
Residual Centralization Factor: 0.0875	**Reward Share:** 0.4112

Table 7.4: Selfish mining results under real-world mining power distribution ($\epsilon = 0$).

Adversarial Share	**Petty-Compliant Mining Pool Shares**
0.07902	[0.29033, 0.24852, 0.12858, 0.11527, 0.03438, 0.03000, 0.02101, 0.05289]
Residual Centralization Factor: 0.1967	**Reward Share:** 0.0794
Adversarial Share	**Petty-Compliant Mining Pool Shares**
0.11527	[0.29033, 0.24852, 0.12858, 0.07902, 0.03438, 0.03000, 0.02101, 0.05289]
Residual Centralization Factor: 0.1968	**Reward Share:** 0.1166
Adversarial Share	**Petty-Compliant Mining Pool Shares**
0.12858	[0.29033, 0.24852, 0.11527, 0.07902, 0.03438, 0.03000, 0.02101, 0.05289]
Residual Centralization Factor: 0.1961	**Reward Share:** 0.1306
Adversarial Share	**Petty-Compliant Mining Pool Shares**
0.24852	[0.29033, 0.12858, 0.11527, 0.07902, 0.03438, 0.03000, 0.02101, 0.05289]
Residual Centralization Factor: 0.1672	**Reward Share:** 0.2980
Adversarial Share	**Petty-Compliant Mining Pool Shares**
0.29033	[0.24852, 0.12858, 0.11527, 0.07902, 0.03438, 0.03000, 0.02101, 0.05289]
Residual Centralization Factor: 0.1453	**Reward Share:** 0.3785

- A valid block is mined on top of adversarial block $B_{\mathcal{A}}$:

$$\texttt{Hash}\big(\texttt{Hash}(B_{\mathcal{A}}), \texttt{MR}, \texttt{nonce}\big) \leq \texttt{Target} \ . \tag{7.20}$$

- The coinbase transaction is included in the list of block transactions, meaning a valid Merkle inclusion `proof` is available to demonstrate that the coinbase transaction tx is a leaf of a Merkle tree with Merkle root `MR`.
- The payout address `add` is included in the coinbase transaction.

If the witness transaction satisfies the conditions above, the bribe `br` will be transferred from the smart contract deposit to the payout address `add`. However, if the adversary submits a witness proving the mining of a block on top of the non-adversarial block $B_{\overline{\mathcal{A}}}$, it can recollect `br`. Upon submission of a valid witness of mining on top of the non-adversarial block $B_{\overline{\mathcal{A}}}$, the bribe will be transferred to the adversarial payout address $\texttt{add}_{\mathcal{A}}$ stored in the contract.

7.A3.10 Duration and Cost of the Initial Loss Period in Selfish Mining

As discussed in the literature [19, 20, 41], all mining destruction attacks, including selfish mining, face an initial period of financial loss before the mining difficulty adjusts. During the first epoch of selfish mining, some adversarial blocks may become orphaned, while the adversarial block generation rate remains unchanged. Therefore, due to the loss of some adversarial blocks, the adversarial mining pool incurs a profit loss before the difficulty adjustment. However, after the difficulty adjustment at the end of the attack's first epoch, the mining difficulty reduces, leading to an increase in the adversarial block generation rate. If this increase in the block generation rate can compensate for

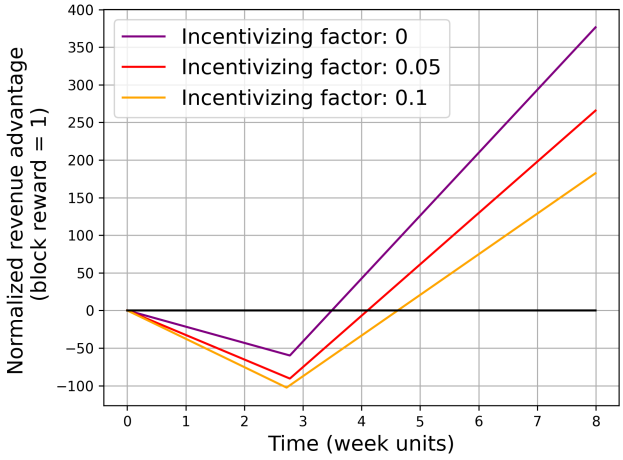


Figure 7.5: The normalized revenue advantage of selfish mining over time (adversarial mining power share: 0.29033).

the loss of orphaned adversarial blocks, the selfish mining attack can become profitable.

In Fig. 7.5, we depict the normalized revenue advantage of the selfish mining attack for a real-world mining pool with mining share 0.29033 as a function of time for different values of incentivizing factor ϵ. The revenue advantage is defined as the difference between the attack revenue and honest mining revenue since the launch of the attack [20]. As shown in Fig. 7.5, during the first epoch of the attack, which lasts longer than the standard 2-week epoch duration, the revenue advantage decreases, indicating a financial loss for the adversarial mining pool. However, after the difficulty adjustment at the end of the first epoch, the revenue advantage of the selfish mining attack begins to increase. A couple of weeks after the attack begins, the revenue advantage finally turns positive, meaning that selfish mining can be considered profitable from that point onward. The initial duration in which the revenue advantage is negative, referred to as the profit lag in [20], can be considered the initial loss duration of a selfish mining attack. As can be seen in Fig. 7.5, the initial cost of the attack increases with a higher incentivizing factor. This is because, in an environment with a higher incentivizing factor, the petty-compliant pools are more likely to behave honestly, and consequently, the adversary must pay a higher bribe to incentivize them.

7.A4 Bribery Attack: Supplementary

7.A4.1 Proof of Thm. 7.2

Proof. Assume $L = 2016$ denotes the epoch length, R denotes the block reward, and λ denotes the block mining rate (the number of blocks mined per unit of time) when all the mining pools follow the honest strategy. We first calculate the time-averaged profit of the adversarial pool under the bribery attack. Since the average number of target blocks per epoch for which a rival block is mined is equal to k, we can conclude that the average number of non-adversarial blocks that get orphaned per epoch is also k. Given that, on average, the same number of non-adversarial blocks are orphaned in each epoch, the mining difficulty is well-adjusted, indicating that the mining rate in all epochs is the same as λ. Therefore, the DURATION of each epoch is equal to L/λ. In the absence of an attack, out of L canonical blocks per epoch, αL blocks are adversarial. Under the bribery attack, however, k blocks are orphaned, where all of them are non-adversarial. The k non-adversarial orphaned blocks need to be replaced with k new blocks, of which αk blocks are adversarial. This implies that under the bribery attack, the number of adversarial blocks per epoch increases to $\alpha(L+k)$ blocks. Therefore, the adversarial mining pool receives a total REV of $\alpha_{\mathcal{A}}(L+k)R$ per epoch. The cost that the adversary incurs in each epoch includes the mining cost and the bribery cost. Here, we ignore the mining cost in our calculation as it is the same as when mining honestly. The total bribery COST in each epoch is $k\mathtt{br}R$ since the adversary needs to pay a normalized bribe of $\mathtt{br}$ for each of the k successful attempts of the bribery attack. Therefore, the time-averaged profit of the bribery attack can be obtained as follows:

$$\textsc{Profit}_{\mathtt{br}} = \frac{\textsc{Rev} - \textsc{Cost}}{\textsc{duration}} = \frac{\lambda\big(\alpha_{\mathcal{A}}(L+k)R - k\mathtt{br}R\big)}{L} = \lambda\alpha_{\mathcal{A}}R + \frac{\lambda kR}{L}(\alpha_{\mathcal{A}} - \mathtt{br}) \ . \tag{7.21}$$

The time-averaged profit of honest mining is equal to $\textsc{Profit}_{\mathcal{H}} = \lambda\alpha_{\mathcal{A}}R$. The bribery attack can strongly dominate the honest strategy if the following inequalities hold:

$$\textsc{Profit}_{\mathtt{br}} > \textsc{Profit}_{\mathcal{H}} \iff \lambda\alpha_{\mathcal{A}}R + \frac{\lambda kR}{L}(\alpha_{\mathcal{A}} - \mathtt{br}) > \lambda\alpha_{\mathcal{A}}R \iff \alpha_{\mathcal{A}} > \mathtt{br} \ . \tag{7.22}$$

□

7.A4.2 Bribery Attack under the Assumption of Unknown Miners

In Sect. 7.4.1, we analyzed the bribery attack under the assumption that the miner of a target block is known. However, a mining pool may choose not to reveal its identity in the coinbase transaction. This makes it difficult for the adversarial and remaining petty-compliant mining pools to detect the miner of the target block. In this section, we analyze the bribery attack in the setting where the miner of the target block is unknown. In the following, we only discuss the smart contract-based attack.

7.A4.2.1 Bribery Attack Using Smart Contracts.

An adversarial mining pool with mining share $\alpha_{\mathcal{A}}$ can conduct the smart contract-based bribery attack if there is at least one petty-compliant mining pool $p_j \in \mathcal{P}$ whose residual centralization factor is less than the mining share of the adversarial mining pool. Otherwise, the attack is not feasible.

Assume that block B_1 represents the head of the canonical chain, and its corresponding miner is not known to the adversarial and petty-compliant mining pools. Block B_1 is considered a target block of the bribery attack if it is a non-adversarial block. Let β_j denote the residual centralization factor with respect to the mining pool p_j, and $\texttt{min}(\beta_j)$ denote the minimum residual centralization factor among all the mining pools. The adversarial mining pool deposits two normalized bribes, $\texttt{br}_1$ and $\texttt{br}_2 = \epsilon$, in the smart contract, where $\texttt{br}_1$ can take any value in the range $[\texttt{min}(\beta_j) + \epsilon, \alpha_{\mathcal{A}} - 2\epsilon)$. The other details are similar to the smart contract-based attack in the known miner setting.

Lemma 7.6. Assume block B_1 which denotes the head of the canonical chain is mined by an unknown mining pool. Assume further that block B_1 is a target of the smart contract-based bribery attack that offers two normalized bribes $\texttt{br}_1$ and $\texttt{br}_2 = \epsilon$. In an environment with the incentivizing factor ϵ, a petty-compliant mining pool p_j which is not the miner of block B_1 is incentivized to follow the bribery attack strategy if $\texttt{br}_1 \geq \beta_j + \epsilon$, where β_j denotes the residual centralization factor w.r.t. the mining pool p_j.

Proof. Let $\mathcal{P}$ denote the set of all mining pools (both semi-rational and adversarial). If a petty-compliant mining pool $p_j \in \mathcal{P}$ mines on top of the target block, its expected return for the next block is equal to $r_1 = \alpha_j R$. If it tries to mine a rival block, its expected return is equal to $r_2 = \alpha_j p_{\texttt{success}}(1 + \texttt{br}_1)R + \alpha_j(1 - p_{\texttt{success}})\texttt{br}_1 R$, where $p_{\texttt{success}}$ represents the probability that the mining pool p_j wins the fork race. Let $\mathcal{P}_{\bar{j}}$ denote the set of all the mining pools excluding p_j. The target block B_i can be mined by

any of the mining pools in $\mathcal{P}_{\overline{j}}$. If block B_1 is mined by mining pool p_i, p_j will win the fork race with probability $1 - \alpha_i$ as all the remaining mining pools are incentivized to mine on top of the rival block. Therefore, the total probability that the mining pool p_j wins the fork race can be obtained as follows:

$$p_{\texttt{success}} = \frac{\sum_{\mathcal{P}_j} \alpha_i(1-\alpha_i)}{1-\alpha_j} = 1 - \beta_j, \tag{7.23}$$

where β_j is the residual centralization factor w.r.t. the mining pool p_j. The mining pool p_j is incentivized to follow the adversarial bribing strategy if $r_2 \geq r_1 + \epsilon\alpha_j R$. We have:

$$\begin{aligned} &r_2 \geq r_1 + \epsilon\alpha_j R \iff \alpha_j(1-\beta_j)(1+\texttt{br}_1)R + \alpha_j\beta_j\texttt{br}_1 R \geq \alpha_j R + \epsilon\alpha_j R \\ &\iff \texttt{br}_1 \geq \beta_j + \epsilon \ . \end{aligned} \tag{7.24}$$

□

Lemma 7.7. In an environment with an unknown miner setting and an incentivizing factor ϵ, an adversarial mining pool with mining share $\alpha_{\mathcal{A}}$ is incentivized to conduct a smart contract-based bribery attack if there exists at least one petty-compliant mining pool such as p_j with residual centralization factor β_j for which the following inequality holds:

$$\alpha_{\mathcal{A}} > \beta_j + 2\epsilon \ . \tag{7.25}$$

Proof. According to Thm. 7.2, the bribery attack can be profitable as long as the bribe that the adversarial mining pool pays out for orphaning a non-adversarial block is less than its mining share. Therefore, since $\alpha_{\mathcal{A}} > \beta_j + 2\epsilon$, the adversarial mining pool can set the total bribe for orphaning a single non-adversarial block to $\texttt{br} = \beta_j + 2\epsilon$ and still profits from the bribery attack. According to Lemma 7.6, with a bribe of value $\texttt{br}$, the petty-compliant mining pool p_j is incentivized to follow the adversarial bribing strategy. Since there exists at least one petty-compliant mining pool that follows the adversarial strategy, the attack is profitable. □

7.A4.3 Bribery Attack Using Whale Transactions

In the bribery attack using whale transactions, the adversarial mining pool sets its bribe as the transaction fee of a transaction that is only valid if included in the chain mined on top of the rival block. This transaction becomes invalid if included in the chain mined on top of the target block. We first explain the attack.

7.A4.3.1 Attack Description

Let $\alpha_{\mathcal{A}}$ represent the mining share of the adversarial mining pool $p_{\mathcal{A}}$. Assume block B_1 denotes the head of the canonical chain and is mined by mining pool p_i with mining share α_i. From the perspective of the adversarial mining pool, block B_1 can be considered a valid target block for the bribery attack if the following condition holds:

- B_1 is a non-adversarial block.
- $\alpha_{\mathcal{A}} > \frac{\alpha_i}{1-\alpha_i} + \epsilon(\frac{1}{1-\alpha_i} + 1)$.

If the conditions above are not satisfied, the attack is not applicable, and the adversarial mining pool waits for the next block. If block B_1 is a valid target block, the adversarial mining pool proceeds with the bribery attack.

We assume an adversarial transaction $tx_{\mathcal{A}}$ is already included in block B_1. The adversarial mining pool then creates two bribes, $\texttt{br}_1$ and $\texttt{br}2$, and sets them as the transaction fees for transactions tx_1 and tx_2, respectively. These transactions are only valid if $tx_{\mathcal{A}}$ is not included beforehand in the canonical chain. The adversarial mining pool publishes transactions tx_1 and tx_2 once block B_1 is published. $\texttt{br}_1$ serves as a bribe for the miner of the rival block. $\texttt{br}_2$ serves as a bribe for the miner of the supporting block, the block that is mined on top of the rival block in the case of a fork race between the rival and target blocks. Once transactions tx_1 and tx_2 are published in the network, a petty-compliant mining pool should choose one of the following two strategies: it can either continue mining on top of block B_1 without including transaction tx_1, or it can try to mine a rival block that includes transaction tx_1[8]. If the next block B_2 is mined on top of block B_1, the bribery attack targeting B_1 fails, and the adversarial mining pool repeats the same attack on block B_2. If the next block B_2 is a rival block for B_1, the attack is successful.

During this attack, the adversarial mining pool commits to the following strategy: if no rival block is published, it mines on top of the target block (the head of the canonical chain). Once a rival block is published, it switches to mining on top of the rival block.

Lemma 7.8. Assume block B_1 is mined by mining pool p_i with mining share α_i. Assume further that block B_1 is a target of the whale transaction-based bribery attack that offers two normalized bribes $\texttt{br}_1 = \frac{\alpha_i+\epsilon}{1-\alpha_i}$ and $\texttt{br}_2 = \epsilon$. In an environment with the incentivizing factor ϵ, all the petty-compliant mining pools except p_i are incentivized to mine a rival block for B_1.

[8]The miner of the rival block can include both transactions tx_1 and tx_2 in its block. However, it would be a wise decision to leave tx_2 as a bribe for the supporting block.

Proof. Let $\mathcal{P}_{\bar{i}}$ denote the set of all mining pools excluding p_i. Under the bribery attack, the petty-compliant mining pools need to choose between mining on top of block B_1 or mining on top of its parent block to mine a rival block.

Let $B_{\texttt{next}}$ denote the next block that is mined during the attack. We first determine the best strategy that a non-adversarial mining pool $p_j \in \mathcal{P}_{\bar{i}}$ should take once block $B_{\texttt{next}}$ is published. If block $B_{\texttt{next}}$ is mined on top of B_1, the attack fails and all mining pools, including p_j, will continue mining on top of B_1. However, if $B_{\texttt{next}}$ is mined on top of B_0, a fork race will occur between B_1 and $B_{\texttt{next}}$. During this fork race, since there is a bribe of $\texttt{br}_2 = \epsilon$ on top of block $B_{\texttt{next}}$, all the mining pools in $\mathcal{P}_{\bar{i}}$, including p_j, will mine on top of block B_1, and only p_i will mine on top of block B_1.

Knowing the best strategy for pool p_j after the publishing of block $B_{\texttt{next}}$, we can obtain the optimal strategy for p_j to follow before $B_{\texttt{next}}$ is published. To this end, we compare the average expected return under the following two strategies for the next block:

- Strategy π_1: p_j mines on top of block B_1 and does not include any of the transactions tx_1 or tx_2 in the block, as these transactions are not valid in a chain mined on top of block B_1.
- Strategy π_2: p_j mines on top of block B_0 and includes only tx_1 with a transaction fee of $\texttt{br}_1 R$ in the block. In this strategy, the mining pool p_j refrains from including tx_2 in its block, leaving it as a bribe on top of its potential block in the upcoming fork race.

Under strategy π_1, the expected return of mining pool p_j for the next block is equal to $r_1 = \alpha_j R$. Under strategy π_2, the expected return of mining pool p_j for the next block can be obtained as follows:

$$r_2 = \alpha_j \cdot R_{\texttt{rival}} \cdot p_{\texttt{success}} \ , \tag{7.26}$$

where $R_{\texttt{rival}}$ is the total block reward of the rival block, and $p_{\texttt{success}}$ is the probability that the rival block wins the fork race against the target block B_1. The total block reward $R_{\texttt{rival}}$ consists of both the block reward and the transaction fee of tx_1, implying that $R_{\texttt{rival}} = (1 + \texttt{br}_1)R$. The probability that the rival block wins the fork race is $1-\alpha_i$, as the only mining pool mining on top of the target block is p_i. All the remaining petty-compliant mining pools choose to mine on top of the rival block due to the bribe $\texttt{br}_2 = \epsilon$ available on top of it. Additionally, the adversarial mining pool mines on top of the rival block since it has already committed to it. Therefore, we obtain $r_2 = \alpha_j(1 - \alpha_i)(1 + \texttt{br}_1)R$. In an environment with incentivizing factor ϵ, mining pool p_j chooses the strategy π_2 over strategy π_1 if $r_2 \geq r_1 + \epsilon\alpha_j R$. We have:

$$r_2 \geq r_1 + \epsilon\alpha_j R \iff \alpha_j(1-\alpha_i)R(1+\texttt{br}_1) \geq \alpha_j R + \epsilon\alpha_j R \iff \texttt{br}_1 \geq \frac{\alpha_i + \epsilon}{1-\alpha_i} \ . \tag{7.27}$$

□

Theorem 7.4. In an environment with an incentivizing factor ϵ, an adversarial mining pool with mining share $\alpha_{\mathcal{A}}$ is incentivized to conduct a whale transaction-based bribery attack on any target block B if block B is mined by a mining pool with mining share α_i, where $\alpha_{\mathcal{A}} > \frac{\alpha_i}{1-\alpha_i} + \epsilon\left(\frac{1}{1-\alpha_i} + 1\right)$.

Proof. According to Lemma 7.8, in an environment with an incentivizing factor ϵ, any block B mined by a mining pool with mining share α_i is susceptible to a whale transaction-based bribery attack if normalized bribes are set as $\mathtt{br}_1 = \frac{\alpha_i+\epsilon}{1-\alpha_i}$ and $\mathtt{br}_2 = \epsilon$. Therefore, to conduct a bribery attack on block B, the adversarial mining pool should pay the normalized bribe of $\mathtt{br} = \mathtt{br}_1 + \mathtt{br}_2 = \frac{\alpha_i}{1-\alpha_i} + \epsilon\left(\frac{1}{1-\alpha_i} + 1\right)$. According to Thm. 7.2, the bribery attack is profitable for the adversarial mining pool if $\alpha_{\mathcal{A}} > \mathtt{br}$. This completes the proof. □

According to Thm. 7.4, assuming $\epsilon = 0$, an adversarial mining pool with a mining share $\alpha_{\mathcal{A}}$ can perform a bribery attack on blocks mined by pools whose mining power is less than $\frac{\alpha_{\mathcal{A}}}{1+\alpha_{\mathcal{A}}}$. This implies that even mining pools with mediocre or low amounts of mining power can exploit weaker mining pools through the bribery attack.

7.A4.4 Proof of Thm. 7.3

To prove Thm. 7.3, we first present the following lemma.

Lemma 7.9. Assume block B_1 is mined by mining pool p_i with mining share α_i. Assume further that block B_1 is a target of the smart contract-based bribery attack that offers two normalized bribes $\mathtt{br}_1 = \alpha_i + \epsilon$ and $\mathtt{br}_2 = \epsilon$. In an environment with the incentivizing factor ϵ, all the petty-compliant mining pools except p_i are incentivized to follow the bribery attack strategy.

Proof. Let $\mathcal{P}_{\bar{i}}$ denote the set of all mining pools excluding p_i. Under the bribery attack, the petty-compliant mining pools need to choose between mining on top of block B_1 or mining on top of its parent block to mine a rival block.

Let $B_{\mathtt{next}}$ denote the next block that is mined during the attack. We first determine the best strategy that a non-adversarial mining pool $p_j \in \mathcal{P}_{\bar{i}}$ should take once block $B_{\mathtt{next}}$ is published. If block $B_{\mathtt{next}}$ is mined on top of B_1, the attack fails and all mining pools, including p_j, will continue mining on top of $B_{\mathtt{next}}$. However, if $B_{\mathtt{next}}$ is mined on top of B_0, a fork race will occur between B_1 and $B_{\mathtt{next}}$. During this fork race, since there is a bribe of $\mathtt{br}_2 = \epsilon$ on top of block $B_{\mathtt{next}}$, all the mining pools in $\mathcal{P}_{\bar{i}}$, including p_j, will mine on top of block

B_1, and only p_i will mine on top of block B_1. Note that even for the adversarial pool, mining on top of $B_{\texttt{next}}$ weakly dominates mining on top of B_1. If the adversarial mining pool manages to mine the next block on top of $B_{\texttt{next}}$, the bribe $\texttt{br}_2 = \epsilon$ will be cleared by the adversary, and there is no need to pay that.

Knowing the best strategy for mining pool $p_j \in \mathcal{P}_{\bar{i}}$ after the publishing of block $B_{\texttt{next}}$ is to mine on top of that, we can obtain the optimal strategy for p_j to follow before $B_{\texttt{next}}$ is mined. To this end, we compare the average expected return under the following two strategies for the next block:

- Strategy π_1: p_j mines on top of block B_1.
- Strategy π_2: p_j mines on top of block B_0.

Under strategy π_1, the expected return of mining pool p_j for the next block is equal to $r_1 = \alpha_j R$. Under strategy π_2, the expected return of mining pool p_j for the next block can be obtained as follows:

$$r_2 = \alpha_j R_{\texttt{success}} p_{\texttt{success}} + \alpha_j R_{\texttt{failure}} \cdot p_{\texttt{failure}} \ . \tag{7.28}$$

The expected return is composed of two parts: the return when the mining pool p_j wins the fork race against target block B_1, and the return when the mining pool p_j loses the fork race. According to the proof of Lemma 7.8, the expected return under winning the fork race is equal to $R_{\texttt{success}} = \alpha_j(1-\alpha_i)(1+\texttt{br}_1)R$. However, if the mining pool p_j loses the fork race, despite losing the block reward, it can collect the bribe deposited in the smart contract as it has mined a valid rival block for block B_1. Therefore, $R_{\texttt{failure}} = \texttt{br}_1 R$. As a result, the total expected return can be obtained as follows:

$$r_2 = \alpha_j(1-\alpha_i)(1+\texttt{br}_1)R + \alpha_j\alpha_i\texttt{br}_1 R \ . \tag{7.29}$$

In an environment with incentivizing factor ϵ, mining pool p_j follows the bribery attack strategy over the honest strategy if $r_2 \geq r_1 + \epsilon\alpha_j R$. We have:

$$\begin{aligned} r_2 \geq r_1 + \epsilon\alpha_j R &\iff \alpha_j(1-\alpha_i)(1+\texttt{br}_1)R + \alpha_j\alpha_i\texttt{br}_1 R \geq \alpha_j R + \epsilon\alpha_j R \\ &\iff \texttt{br}_1 \geq \alpha_i + \epsilon \ . \end{aligned} \tag{7.30}$$

□

Proof of Thm. 7.3. According to Lemma 7.9, in an environment with an incentivizing factor ϵ, any block B mined by a mining pool with mining share α_i is susceptible to a smart contract-based bribery attack if normalized bribes are set as $\texttt{br}_1 = \alpha_i + \epsilon$ and $\texttt{br}_2 = \epsilon$. Therefore, to conduct a bribery attack on block B, the adversarial mining pool should pay the normalized bribe of $\texttt{br} = \texttt{br}_1 + \texttt{br}_2 = \alpha_i + 2\epsilon$. According to Thm. 7.2, the bribery attack is profitable for the adversarial mining pool if $\alpha_{\mathcal{A}} > \texttt{br}$. This completes the proof. □

7.A4.5 Markov Chain Analysis of Bribery Attack

To analyze the bribery attack, we divide the mining pools into three groups: the adversarial mining pool, the target mining pools, and the non-target mining pools. The target mining pools are those whose blocks are considered targets for the bribery attack. The non-target mining pools are those whose blocks are not considered targets for the bribery attack. We denote by $\alpha_{\mathcal{A}}$ the mining share of the adversarial pool. Let $\mathcal{P}_{\texttt{target}}$ denote the set of N target mining pools. For each target mining pool $p_i \in \mathcal{P}_{\texttt{target}}$, we denote its mining share by b_i. We denote by b the total mining share of all the target mining pools in $\mathcal{P}_{\texttt{target}}$. We define value β as follows:

$$\beta = \sum_{p_i \in \mathcal{P}_{\texttt{target}}} \frac{b_i}{1 - b_i} \tag{7.31}$$

States: State S_0 denotes the state in which the tip of the chain is not a target block and all the pools are mining on the tip of the chain. State S_1^i for $i \in [N]$[9] denotes a state in which the tip of the chain is a target block mined by the mining pool p_i and the adversary has placed 2 normalized bribes $\texttt{br}_1 = b_i + \epsilon$ and $\texttt{br}_2 = \epsilon$ for orphaning the chain tip. According to Thm. 7.3, these bribes incentivizes all the petty-compliant mining pools except for the target pool p_i to mine a rival block. State S_2^i for $i \in [N]$ denotes a state in which a rival block is mined for the target block of mining pool p_i. Let P_0, P_1^i, and P_2^i, for $i \in [N]$, denote the probability of being at states S_0, S_1^i, and S_2^i, respectively.

We assume the smart contract of the bribery attack is designed such that $\texttt{br}_1$ is payable only upon mining a rival block, and $\texttt{br}_2$ is payable only upon mining a non-adversarial block supporting the rival block. In all other cases, the adversarial mining pool is able to clear the bribes.

State transitions:

- Let S_0 be the current state. If the adversarial mining pool mines a new tip block, it immediately publishes the block, and all the mining pools accept the block. The system remains in the same state.
- Let S_0 be the current state. If a non-target mining pool mines and publishes a new tip block, all the mining pools accept the block. The system remains in the same state.
- Let S_0 be the current state. If a target mining pool p_i mines a new tip block $B_1^{\texttt{target}}$, this block is considered a target for the bribery attack. The adversarial mining pool places two normalized bribes $\texttt{br}_1 = b_i + \epsilon$ and $\texttt{br}_2 = \epsilon$ for orphaning $B_1^{\texttt{target}}$. The system transitions to state S_1^i. At the new state S_1^i, mining pool p_i and the adversarial mining pool mine on top

[9] We define $[N]$ to be $\{1, 2, \ldots, N\}$.

of block $B_1^{\texttt{target}}$, while the other mining pools mine on top of the parent of block $B_1^{\texttt{target}}$ to mine a rival block.

- Let S_1^i be the current state. If the adversarial mining pool mines the next block, which is a block that extends $B_1^{\texttt{target}}$, it immediately publishes the block and clears the previous bribes. In this case, all mining pools adopt the chain ending at the adversarial block, and the system transitions to state S_0.
- Let S_1^i be the current state. If any petty-compliant pool except p_i mines the next block, which is a rival block to $B_1^{\texttt{target}}$, it immediately publishes the block and collects the bribe $\texttt{br}_1 = b_i + \epsilon$. The system transitions to state S_2^i. At this new state S_2^i, all the mining pools except p_i mine on the rival block.
- Let S_1^i be the current state. If mining pool p_i mines the next block $B_2^{\texttt{target}}$, which extends $B_1^{\texttt{target}}$, all the mining pools accept the chain ending at the previous target block $B_1^{\texttt{target}}$, and the adversary clears the previous bribes. The adversarial mining pool places new bribes for orphaning the new target block $B_2^{\texttt{target}}$. The system remains in the same state.
- Let S_2^i be the current state. If the adversarial mining pool mines the next block, which is a block that extends the rival block, the adversary immediately publishes the block and clears the second bribe $\texttt{br}_2 = \epsilon$. All the mining pools adopt the rival fork ending at the adversarial block[10], and the system transitions to state S_0.
- Let S_2^i be the current state. If a non-target mining pool mines the next block, which is a block that extends the rival block, it publishes the block and collects the bribe $\texttt{br}_2 = \epsilon$. All the mining pools adopt the rival fork ending at the non-target block, and the system transitions to state S_0.
- Let S_2^i be the current state. If a target mining pool p_j, where p_j differs from p_i, mines the next block $B_2^{\texttt{target}}$, which is a block that extends the rival block, it publishes the block and collects the bribe $\texttt{br}_2 = \epsilon$. All the mining pools adopt the rival fork ending at the rival block. The adversarial mining pool places new bribes for orphaning the new target block $B_2^{\texttt{target}}$, and the system transitions to state S_1^j.
- Let S_2^i be the current state. If the target mining pool p_i mines the next block $B_2^{\texttt{target}}$, which is a block that extends the target block $B_1^{\texttt{target}}$, it publishes the block, and the adversarial mining pool clears the bribe $\texttt{br}_2 = \epsilon$. All the mining pools adopt the target fork ending at the first target block $B_1^{\texttt{target}}$. The adversarial mining pool places new bribes for orphaning the new target block $B_2^{\texttt{target}}$, and the system transitions to state S_1^i.

[10] According to Lemma 7.2, if the mining share of the target mining pool p_i is less than 0.4302, it adopts the longer rival fork

Table 7.5: State transitions under the bribery attack.

Initial State	Next State	Transition Probability	Total Blocks Added	Adversary Expected Profit
S_0	S_0	$\alpha_{\mathcal{A}}$	1	1
	S_0	$1-b-\alpha_{\mathcal{A}}$	1	0
	S_1^i	b_i	0	0
S_1^i	S_0	$\alpha_{\mathcal{A}}$	2	1
	S_1^i	b_i	1	0
	S_2^i	$1-\alpha_{\mathcal{A}}-b_i$	0	$-b_i-\epsilon$
S_2^i	S_0	$\alpha_{\mathcal{A}}$	2	1
	S_0	$1-b-\alpha_{\mathcal{A}}$	2	$-\epsilon$
	S_1^i	b_i	1	0
	S_1^j $(j \neq i)$	b_j	1	$-\epsilon$

The state transition probabilities, the expected profit of the adversarial pool for each state transition, and the number of blocks added for each state transition are described in Table 7.5.

Based on the transition probabilities mentioned in Table 7.5, we can obtain the state probabilities as follows:

$$\begin{aligned} P_0 &= \frac{1-b+\alpha_{\mathbf{A}}\beta}{1+\beta} \\ P_1^i &= \frac{b_i}{(1-b_i)(1+\beta)} \quad , \\ P_2^i &= \frac{b_i(1-\alpha_{\mathbf{A}}-b_i)}{(1-b_i)(1+\beta)} \quad . \end{aligned} \tag{7.32}$$

According to Table 7.5, the reward share $R_{\mathcal{A}}$ of the adversarial pool under the bribery attack can be obtained as follows:

$$R_{\mathcal{A}} = \frac{\alpha_{\mathcal{A}} - \sum_{i=1}^{N}\Big(P_1^i(1-\alpha_{\mathcal{A}}-b_i)(b_i+\epsilon) + P_2^i(1-\alpha_{\mathcal{A}}-b_i)\epsilon\Big)}{P_0(1-b) + \sum_{i=1}^{N}\Big(P_1^i(b_i+2\alpha_{\mathcal{A}}) + P_2^i(2-b)\Big)} \quad . \tag{7.33}$$

7.A4.6 Markov Chain Analysis of Undercutting Attack

To analyze the undercutting attack, we divide the mining pools into three groups: the adversarial mining pool, the target mining pools, and the non-target mining pools. The target mining pools are those whose blocks are considered targets for the undercutting attack. The non-target mining pools are those

whose blocks are not considered targets for the undercutting attack. We use the same notations as in App. 7.A4.5. Whenever a target block is mined, the adversarial mining pool attempts to undercut the target block. If the adversary successfully mines a rival block, it places a bribe of ϵ on top of the rival block to incentivize all mining pools, except the miner of the target block, to mine on top of the rival block.

States: State S_0 denotes the state in which the tip of the chain is not a target block and all the pools are mining on the tip of the chain. State S_1^i for $i \in [N]$ denotes a state in which the tip of the chain is a target block mined by the mining pool p_i, and the adversarial mining pool is attempting to undercut the chain tip. State S_2^i for $i \in [N]$ denotes a state in which the adversary has mined a rival block for the target block of mining pool p_i and placed a bribe of $\texttt{br} = \epsilon$ on top of the rival block. Let P_0, P_1^i, and P_2^i, for $i \in [N]$, denote the probability of being at states S_0, S_1^i, and S_2^i, respectively.

State Transitions:

- Let S_0 be the current state. If the adversarial mining pool mines a new tip block, it immediately publishes the block, and all mining pools accept it. The system remains in the same state.
- Let S_0 be the current state. If a non-target mining pool mines and publishes a new tip block, all mining pools accept it. The system remains in the same state.
- Let S_0 be the current state. If a target mining pool p_i mines a new tip block $B_1^{\texttt{target}}$, this block becomes the target for the undercutting attack. The system transitions to state S_1^i. At S_1^i, all non-adversarial mining pools mine on top of $B_1^{\texttt{target}}$, while the adversarial mining pool mines on the parent block of $B_1^{\texttt{target}}$ to create a rival block.
- Let S_1^i be the current state. If the adversarial mining pool mines a rival block for $B_1^{\texttt{target}}$, it immediately publishes the block and places a bribe $\texttt{br} = \epsilon$ on top of it. The system transitions to state S_2^i. At S_2^i, all mining pools except p_i mine on top of the rival block.
- Let S_1^i be the current state. If a non-target mining pool mines a block extending $B_1^{\texttt{target}}$, it publishes the block, and all mining pools adopt the chain ending at this non-target block. The system transitions to state S_0.
- Let S_1^i be the current state. If a target mining pool p_j, where $p_j \neq p_i$, mines a block $B_2^{\texttt{target}}$ extending $B_1^{\texttt{target}}$, it publishes the block. The adversarial mining pool then adopts $B_1^{\texttt{target}}$ and attempts to undercut $B_2^{\texttt{target}}$. The system transitions to state S_1^j.
- Let S_1^i be the current state. If the target mining pool p_i mines a block $B_2^{\texttt{target}}$ extending $B_1^{\texttt{target}}$, it publishes the block. The adversarial mining pool continues to target $B_2^{\texttt{target}}$ while attempting to undercut it. The system remains in state S_1^i.

Table 7.6: State transitions under the undercutting attack.

Initial State	Next State	Transition Probability	Total Blocks Added	Adversary Expected Profit
S_0	S_0	$\alpha_{\mathcal{A}}$	1	1
	S_0	$1-b-\alpha_{\mathcal{A}}$	1	0
	S_1^i	b_i	0	0
S_1^i	S_2^i	$\alpha_{\mathcal{A}}$	0	0
	S_0	$1-b-\alpha_{\mathcal{A}}$	2	0
	$S_1^j (j \in [N])$	b_j	1	0
S_2^i	S_0	$\alpha_{\mathcal{A}}$	2	2
	S_0	$1-b-\alpha_{\mathcal{A}}$	2	$1-\epsilon$
	S_1^i	b_i	1	0
	$S_1^j \ (j \neq i)$	b_j	1	$1-\epsilon$

- Let S_2^i be the current state. If the adversarial mining pool mines a block extending the rival block, it immediately publishes the block and clears the bribe $\mathtt{br} = \epsilon$. All mining pools adopt the rival fork, and the system transitions to state S_0.
- Let S_2^i be the current state. If a non-target mining pool mines a block extending the rival block, it publishes the block and collects the bribe $\mathtt{br} = \epsilon$. All mining pools adopt the rival fork, and the system transitions to state S_0.
- Let S_2^i be the current state. If a target mining pool p_j, where $p_j \neq p_i$, mines a block $B_2^{\mathtt{target}}$ extending the rival block, it publishes the block and collects the bribe $\mathtt{br} = \epsilon$. The adversarial mining pool adopts $B_1^{\mathtt{target}}$ and attempts to undercut $B_2^{\mathtt{target}}$. The system transitions to state S_1^j.
- Let S_2^i be the current state. If the target mining pool p_i mines a block $B_2^{\mathtt{target}}$ extending $B_1^{\mathtt{target}}$, it publishes the block. The adversarial mining pool clears the bribe $\mathtt{br} = \epsilon$, adopts $B_1^{\mathtt{target}}$, and attempts to undercut $B_2^{\mathtt{target}}$. The system transitions to state S_1^i.

The state transition probabilities, the expected profit of the adversarial pool for each state transition, and the number of blocks added for each state transition are described in Table 7.6. Based on the transition probabilities mentioned in Table 7.6, we can obtain the state probabilities as follows:

$$\begin{aligned} P_0 &= 1 - b(1 + \alpha_{\mathcal{A}}) \\ P_1^i &= b_i \ , \\ P_2 &= \alpha_{\mathcal{A}} b_i \ . \end{aligned} \tag{7.34}$$

According to Table 7.6, the reward share $R_{\mathcal{A}}$ of the adversarial pool under the undercutting attack can be obtained as follows:

$$R_{\mathcal{A}} = \frac{P_0\alpha_{\mathcal{A}} + \sum_{i=1}^{N}\left(P_2^i(2\alpha_{\mathcal{A}} + (1-\alpha_{\mathcal{A}} - b_i)(1-\epsilon))\right)}{P_0(1-b) + \sum_{i=1}^{N}\left(P_1^i(2-2\alpha_{\mathcal{A}} - b) + P_2^i(2-b)\right)} \ . \tag{7.35}$$

7.A4.7 Naive Solutions to Mitigate the Bribery Attack

The introduction of the bribery attack raises the question of how a petty-compliant mining pool can defend against this attack. As discussed in the previous section, conducting a bribery attack requires a higher budget in a setting with unknown miners. This suggests that if a petty-compliant mining pool does not reveal its identity in the block, the probability of being targeted by a bribery attack decreases. This strategy is particularly useful for smaller mining pools, which are more susceptible to bribery attacks. However, not disclosing its identity in the coinbase transaction does not guarantee complete concealment for the mining pool. For instance, petty-compliant mining pools may conduct network analyses to identify the miner of a targeted block. Moreover, if the remaining petty-compliant pools, especially the largest ones, continue to reveal their identity in their blocks, it becomes easier to detect whether a block is mined by one of the largest pools or by a smaller pool prone to the bribery attack.

Another potential solution for petty-compliant mining pools is to engage in commitment games. To protect their blocks, they can submit commitments offering incentives to mining pools that support their block in the event of a fork race. However, this approach ties Bitcoin's security to commitments that typically occur outside the Bitcoin framework. Furthermore, the presence of such commitments may encourage other petty-compliant mining pools to initiate fork races to access the funds deposited in those commitments. For instance, other petty-compliant mining pools may intentionally disrupt block propagation in the network to increase the likelihood of a fork race occurring.

7.A4.8 Smart Contract-Based vs. Whale Transaction-Based Bribery Attacks

One of the advantages of the smart contract-based bribery attack is that there is no need for the adversarial mining pool to commit to its strategy to incentivize the petty-compliant mining pools to follow the attack strategy.

In the whale transaction-based attack, the mining pool p_j must trust the assumption that the adversarial mining pool will mine on top of the rival block

rather than the target block in the case of a fork race. This assumption is necessary as the dominant strategy for the adversarial mining pool is to mine on top of the target block. This is because once a rival block is mined and a fork race is created, the attack is considered successful from the perspective of the adversarial mining pool regardless of the fork race outcome, as one of the blocks will be orphaned. In this case, mining on top of the target block is the dominant strategy for the adversarial mining pool because it can invalidate the bribe transaction in the rival block and avoid paying the bribe. This illustrates that if the adversarial mining pool does not commit to the strategy of mining on top of the rival block in the whale transaction-based attack, the total mining share that mines on top of the targeted block becomes equal to $\alpha_i + \alpha_{\mathcal{A}}$, resulting in unprofitability of the attack from the perspective of the petty-compliant mining pool p_j.

However, in the smart contract-based attack, the adversarial mining pool is not incentivized to mine on top of the target block in the case of a fork race. This is because the bribe $\mathtt{br}_1$ will be paid to the miner of the rival block regardless of the fork race result. Therefore, in contrast to the whale transaction-based attack, there is no hope for the adversarial mining pool to reclaim the bribe $\mathtt{br}_1$ by mining on top of the target block. Also, regarding the bribe $\mathtt{br}_2$, the adversarial mining pool has no preference between the rival block and the target block to mine on top of. If the adversarial mining pool mines on top of the target block, no petty-compliant mining pool can withdraw the bribe $\mathtt{br}_2$ from the smart contract. If it mines on top of the rival block, the adversarial mining pool will be eligible to withdraw $\mathtt{br}_2$ from the smart contract. Therefore, in both scenarios, the adversarial mining pool can reclaim $\mathtt{br}_2$. This indicates that in the smart contract-based attack, mining on top of the rival block dominates mining on top of the target block for the adversarial mining pool.

To make mining on top of the rival block strongly dominate mining on top of the target block, the adversarial mining pool can increase the bribe for the rival block from $\mathtt{br}_1 = \alpha_i + \epsilon$ to $\mathtt{br}_1 = \alpha_i + 2\epsilon$, where the petty-compliant miner of the rival block can use the extra bribe of ϵ to incentivize the adversarial mining pool to mine on top of the rival block. To be more precise, the adversarial mining pool deposits $\mathtt{br}_1 = \alpha_i + 2\epsilon$ and $\mathtt{br}_2 = \epsilon$ in the smart contract, where both bribes $\mathtt{br}_1$ and $\mathtt{br}_2$ can be withdrawn by the adversarial mining pool after the expiration of a time lock. During this time lock, if a petty-compliant mining pool mines a valid rival block, it withdraws $\alpha_i + \epsilon$ of bribe $\mathtt{br}_1$, and the rest of bribe $\mathtt{br}_1$ gets deposited as $\mathtt{br}_3 = \epsilon$ in the smart contract without any time lock. The mining pool that mines on top of the rival block is eligible to withdraw both $\mathtt{br}_2$ and $\mathtt{br}_3$, resulting in a total bribe of 2ϵ. In this case, if the adversarial mining pool mines on top of the rival block, it receives 2ϵ. But, if it mines on top of the target block, it would only be able to withdraw $\mathtt{br}_2 = \epsilon$ after the

time lock, indicating that mining on top of the rival block strongly dominates mining on top of the target block.

7.A4.9 The Impact of DAM on the Profitability of Selfish Mining and Bribery Attacks

The profitability of an adversarial mining pool that engages in selfish mining or bribery attacks is mainly due to the destruction of a portion of the network's mining power. This mining power destruction leads to the reduced total effective mining power of the network. To balance the network's throughput, the difficulty adjustment mechanism lowers the mining difficulty to align with the reduced active mining power. As a result of this reduction in mining difficulty, the block generation rate of the adversarial poll increases, thereby enhancing its profitability.

Each Bitcoin epoch is defined as the duration in which a total of $L = 2016$ blocks are added to the canonical chain. At the end of each epoch, the difficulty adjustment mechanism sets the mining difficulty for the upcoming epoch based on the duration of the past epoch. If a bribing or selfish mining attack occurs during an epoch, some blocks become orphaned, leading to an increase in the epoch's duration and a decrease in the canonical block generation rate. As a result, the difficulty adjustment mechanism at the end of the epoch reduces the mining difficulty for the next epoch to restore the canonical block generation rate. Selfish mining and bribery attacks exploit a bottleneck in the difficulty adjustment mechanism, which fails to distinguish between a portion of the mining power being offline or being wasted due to an attack. The current Bitcoin difficulty adjustment mechanism balances the mining difficulty based on the effective mining power, defined as the mining power that extends the canonical chain and does not contribute to any other blocks outside the canonical chain. The advantage of this approach is that it maintains a nearly constant block generation rate and transaction throughput.

However, it is possible to design a difficulty adjustment mechanism that aligns the difficulty based on active mining power, which is defined as all the mining power used to mine both canonical and non-canonical blocks. A difficulty adjustment mechanism that precisely estimates the active mining power can disincentivize adversarial mining pools from conducting selfish or bribery attacks. This is because, no matter how many blocks the adversarial mining pool manages to orphan, the difficulty adjustment mechanism would account for the total active mining power and would not reduce the mining difficulty. This lack of difficulty reduction deters petty-compliant mining pools from engaging in adversarial behaviors such as selfish mining or bribery attacks.

Although the second approach to designing difficulty adjustment mechanisms can be beneficial to a network of petty-compliant miners, it cannot completely mitigate the mining power destruction attacks. First, this type of DAM is ineffective against a Byzantine mining pool that disregards profitability and acts maliciously only to disrupt blockchain progress. For example, a Byzantine mining pool can exploit the active mining power-based DAM to reduce the transaction throughput. Additionally, there is the critical question of how to design a DAM that accurately estimates active mining power. Papers [38] and [41] propose solutions for calculating active mining power by incorporating the counts of both canonical and orphan blocks into the difficulty adjustment mechanism. In these solutions, the difficulty for the next epoch is calculated based on the total number of orphaned and canonical blocks in the previous epoch. Orphan blocks serve as valid proof of mining power loss. However, it is not always possible to provide the DAM with such proof. For instance, if a portion of the mining power is directed toward a platform other than Bitcoin, valid proof of mining power loss cannot be provided unless the Bitcoin blockchain decides to trust external platforms.

7.A5 Mining Power Distraction Attack

In this section, we present an attack referred to as the *mining power distraction attack*, which is capable of destroying a portion of the network's mining power without producing any evidence of mining power destruction.

In the distraction attack, the adversarial mining pool aims to distract a portion of mining power from mining on top of the canonical chain by incentivizing them to mine on another platform. Since the distracted mining power contributes to a work that is only valid outside the Bitcoin framework, one cannot prove the validity of work to the difficulty adjustment mechanism of Bitcoin. Therefore, even a DAM that balances the difficulty with the active mining power cannot disincentivize the payoff-maximizing mining pools from performing the mining power distraction attack.

The general idea behind the attack is that once the adversarial mining pool mines a block under Bitcoin difficulty, denoted by D_1, it does not publish the block to the other mining pools. Instead, it conveys the following message to them: "I have already mined the next block of the canonical chain. If you want to know the contents of this block, you must submit a valid proof of work under difficulty D_2, where $D_2 < D_1$." We will later discuss the details of how to convey this message and why mining pools should trust it. For now, let us assume that the petty-compliant mining pools accept the existence of a hidden block that extends the canonical chain. This situation puts them in a dilemma:

on one hand, if they continue mining on top of the public chain and successfully mine the next block, it will enter a fork race with the adversarial block. On the other hand, if they wish to access the adversarial block, they need to waste a portion of their mining power to solve a puzzle with difficulty D_2. We will demonstrate that by selecting an appropriate value for D_2, the adversarial mining pool can incentivize the petty-compliant mining pools to choose the latter option, thereby wasting some of their mining power on a puzzle defined outside the Bitcoin framework. Our distraction attack can be considered an extension of the blockchain denial-of-service (BDoS) attack introduced in [31]. The description of the BDoS attack and its differences from our distraction attack are presented in 7.A5.3. The BDoS attack primarily affects mining pools whose revenue is only marginally higher than their mining costs and is ineffective against miners with a higher profit-to-loss ratio. In contrast, our distraction attack can disincentivize mining on Bitcoin, even under the assumption that mining costs are negligible.

Attack Description. Let D_1 denote the Bitcoin mining difficulty. Once the adversarial mining pool mines a block, it deploys a smart contract on an external blockchain platform. This smart contract defines a mining puzzle that is similar to the Bitcoin puzzle, with the difference that the puzzle difficulty is D_2, which is less than D_1. We refer to the solution to this easier puzzle as **mini-PoW**. The smart contract is designed so that if anyone submits a valid mini-PoW, the adversarial mining pool should publish its hidden Bitcoin block; otherwise, it will lose a significant deposit. The adversarial mining pool deposits three normalized bribes of values $\mathtt{br}_1$, $\mathtt{br}_2$, and $\mathtt{br}_3$ in the smart contract. If a user submits a mini-PoW and the adversarial mining pool does not reveal the hidden block within a short time (indicating dishonesty of the claim of having a Bitcoin block), the user is eligible to withdraw $\mathtt{br}_1$. If a user submits a mini-PoW and the adversarial mining pool immediately reveals its hidden Bitcoin block (indicating honesty), the user is eligible to withdraw $\mathtt{br}_2$. If a user mines a supporting block on top of the adversarial block during the fork race, the user is eligible to withdraw $\mathtt{br}_3$.

The smart contract stores two sets of parameters. The first set relates to the Bitcoin chain and includes: the hash of the head of the canonical chain from the perspective of petty-compliant mining pools, denoted by H_1; the Bitcoin mining difficulty D_1; and a commitment to the hash of the adversarial hidden block, denoted as $\mathtt{Hash}(H_{\mathcal{A}})$. The second set of parameters is used to construct a mini-PoW puzzle and includes: a random value serving as a parent block hash H_2; a random value serving as a block Merkle root, denoted as $\mathtt{MR}_2$; and a mining difficulty D_2. The values H_2, $\mathtt{MR}_2$, and D_2 are selected randomly to prevent mining pools from using Bitcoin block PoW solutions as a valid mini-PoW solution. The smart contract satisfies the following conditions:

- If a user submits $\texttt{nonce}_2$ that satisfies the inequality $\texttt{Hash}\big(H_2, \texttt{MR}_2, \texttt{nonce}_2\big) \leq \frac{1}{D_2}$ (successfully mines a mini-PoW), a time lock is activated. If, before the expiration of the time lock, the smart contract deployer submits $\texttt{nonce}_1$ and a Merkle root $\texttt{MR}_1$ that satisfies the inequality $\texttt{Hash}\big(H_1, \texttt{MR}_1, \texttt{nonce}_1\big) \leq \frac{1}{D_1}$ (reveals the hidden block[11]), the user can withdraw $\texttt{br}_2$; otherwise, it can withdraw a larger bribe of $\texttt{br}_1$.
- If a user submits a piece of proof for mining a valid supporting block on top of the adversarial block during the fork race, they can withdraw $\texttt{br}_3$. The user should submit H_3, $\texttt{nonce}_3$, and $\texttt{MR}_3$ that satisfy the equality $\texttt{Hash}(H_3) = \texttt{Hash}(H_{\mathcal{A}})$ and the inequality $\texttt{Hash}\big(H_3, \texttt{MR}_3, \texttt{nonce}_3\big) \leq \frac{1}{D_1}$.

Lemma 7.10. Assume the adversarial mining pool has no hidden block to reveal upon the submission of a valid mini-PoW. In an environment with incentivizing factor ϵ, the petty-compliant mining pools are incentivized to mine for a mini-PoW rather than a Bitcoin block if $\texttt{br}_1 \geq \frac{D_2}{D_1}(1+\epsilon)$.

Proof. Let λ_1 denote the Bitcoin mining rate. In this case, the mini-PoW mining rate is equal to $\lambda_2 = \frac{D_1}{D_2}$. The expected return of mining a Bitcoin block for mining pool p_j is equal to $r_1 = \alpha_j \lambda_1 R$. The expected return of mining a mini-PoW for mining pool p_j is equal to $r_2 = \alpha_j \lambda_2 \texttt{br}_1 R$. The mining pool p_j is incentivized to mine a mini-PoW if $r_2 \geq r_1 + \epsilon \alpha_j \lambda_1 R$. We have:

$$r_2 \geq r_1 + \epsilon \alpha_j \lambda_1 R \iff \frac{D_1}{D_2}\texttt{br}_1 \geq 1 + \epsilon \iff \texttt{br}_1 \geq \frac{D_2}{D_1}(1+\epsilon) \ . \tag{7.36}$$

□

Lemma 7.11. Let D_1 denote the Bitcoin mining difficulty. Consider an adversarial mining pool with a mining share $\alpha_{\mathcal{A}}$, which offers a normalized bribe $\texttt{br}$ for mining each mini-PoW of difficulty D_2. Assume k represents the average number of valid mini-PoW solutions mined per epoch during the distraction attack. If no adversarial block is orphaned, the time-averaged profit of the adversarial mining pool under the distraction attack exceeds its profit under the honest strategy for any value of k, as long as $\texttt{br} < \frac{D_2}{D_1}\alpha_{\mathcal{A}}$.

Proof. The proof is similar to the proof of Thm. 7.2 presented in 7.A4.1. Since k mini-PoW solutions are mined per epoch, an expected $\frac{D_2}{D_1}k$ non-adversarial blocks are wasted per epoch, which must be replaced by new blocks. Of these new $\frac{D_2}{D_1}k$ blocks, $\alpha_{\mathcal{A}}\frac{D_2}{D_1}k$ are adversarial. Therefore, the time-averaged profit of

[11] In App. 7.A5.4, we discuss a smart contract that also forces revealing transactions.

the distraction attack can be obtained as follows:

$$\text{PROFIT}_{\texttt{Distraction}} = \frac{\text{REV} - \text{COST}}{\text{DURATION}} = \frac{\lambda\big(\alpha_{\mathcal{A}}(L + \frac{D_2}{D_1}k)R - k\texttt{br}R\big)}{L} = \lambda\alpha_{\mathcal{A}}R + \frac{\lambda kR}{L}(\frac{D_2}{D_1}\alpha_{\mathcal{A}} - \texttt{br}) \ . \tag{7.37}$$

The time-averaged profit of honest mining is equal to $\text{PROFIT}_{\mathcal{H}} = \lambda\alpha_{\mathcal{A}}R$. The distraction attack can strongly dominate the honest strategy if the following inequalities hold:

$$\text{PROFIT}_{\texttt{Distraction}} > \text{PROFIT}_{\mathcal{H}} \iff \lambda\alpha_{\mathcal{A}}R + \frac{\lambda kR}{L}(\frac{D_2}{D_1}\alpha_{\mathcal{A}} - \texttt{br}) > \lambda\alpha_{\mathcal{A}}R$$
$$\iff \frac{D_2}{D_1}\alpha_{\mathcal{A}} > \texttt{br} \ . \tag{7.38}$$

□

According to Lemma 7.10, if the adversary has no hidden Bitcoin block to reveal, it must set the bribe for each mini-PoW to $\texttt{br}_1 = \frac{D_2}{D_1}(1+\epsilon)$ to incentivize miners to comply with the distraction attack. However, Lemma 7.11 shows that for the distraction attack to be profitable for the adversary, the maximum bribe the adversary can offer per mini-PoW is $\frac{D_2}{D_1}\alpha_{\mathcal{A}}$, which is less than $\texttt{br}_1$. Therefore, as expected, if the adversary has no hidden block, the distraction attack is not profitable for him. For the distraction attack to be profitable, upon submission of a valid mini-PoW, the adversary must be able to reveal a valid Bitcoin block. In this case, rather than offering a large bribe of $\texttt{br}_1 = \frac{D_2}{D_1}(1+\epsilon)$, each mini-PoW receives a reward $\texttt{br}_2$, which, according to Lemma 7.11, should be set to $\texttt{br}_2 < \frac{D_2}{D_1}\alpha_{\mathcal{A}}$.

To demonstrate the feasibility of the distraction attack, we need to show that even if the adversary has a hidden Bitcoin block, petty-compliant pools are still incentivized to mine mini-PoWs with a reward $\texttt{br}_2 < \frac{D_2}{D_1}\alpha_{\mathcal{A}}$ instead of Bitcoin mining. To do this, we must compare the expected returns of Bitcoin mining and mini-PoW mining for a petty-compliant mining pool p_i.

7.A5.1 Expected Return Calculations

Let $r_{\texttt{mini-PoW}}$ and $r_{\texttt{Bitcoin}}$ denote the expected returns of petty-compliant pool p_i for mini-PoW mining and Bitcoin mining, respectively, under the assumption that the adversary has a hidden block to reveal. In an ϵ-semi-rational environment, the petty-compliant mining pool p_i is incentivized to

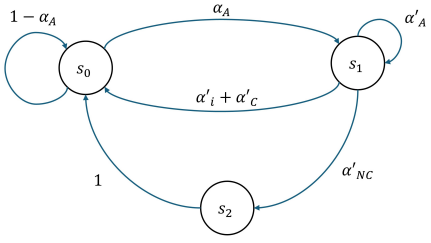
mine a mini-PoW if the following inequality holds: $r_{\texttt{mini-PoW}} \geq r_{\texttt{Bitcoin}} + \epsilon\alpha_i\lambda R$. We define the normalized expected return difference as

$$\Delta = \frac{r_{\texttt{mini-PoW}} - r_{\texttt{Bitcoin}}}{\alpha_i \lambda R} \quad . \tag{7.39}$$

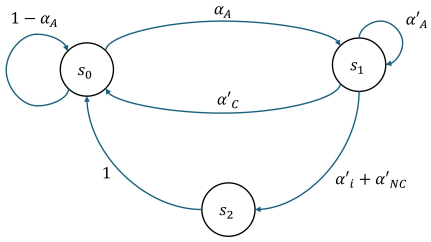
If $\Delta \geq \epsilon$, mining a mini-PoW is the dominant strategy for mining pool p_i. We also define the difficulty ratio d as the ratio of Bitcoin difficulty to mini-PoW difficulty, i.e., $d = \frac{D_1}{D_2}$, where $d \geq 1$ by construction. We consider 4 different mining pools: the adversarial mining pool with mining share $\alpha_{\mathcal{A}}$, the petty-compliant mining pool p_i with mining power α_i that wants to decide which puzzle is more profitable, the set of compliant mining pools with total mining share of α_C that always mines a mini-PoW (if available), and the set of non-compliant mining pools with total mining share of α_{NC} that always mines a Bitcoin block. Note that, under this attack, there is a possibility that some mining pools may decide to stop mining. We do not consider this scenario, as their decision to stop mining would signify the attack's success.

There are 3 different states that can take place: state s_0 where all the mining pools mine on top of the Bitcoin longest chain, state s_1 where the adversarial mining pool has a hidden Bitcoin block and the hash distraction contract is published, and state s_2 where there is a fork race between the adversarial block and a semi-rational block. According to the attack description, there is a bribe of $\texttt{br}_3 = \epsilon$ on top of the adversarial fork in state s_2. The state transitions are depicted in Figures 7.6a and 7.6b. For the sake of calculation simplicity, we assume if the adversarial mining pool mines a block in the state s_1 (mines 2 consecutive blocks), it will publish its first hidden block and repeat the attack with its second block. The petty-compliant pool p_i needs to decide on its action in state s_1: mine a Bitcoin PoW or mine a mini-PoW.

In the first scenario, we consider the case that the petty-compliant pool p_i mines a mini-PoW. The puzzle mining rate at states s_0 and s_2 are the same as λ. However, the puzzle mining rate at state s_1 is equal to $\lambda_1 = \big(d(\alpha_C + \alpha_i) + \alpha_{NC} + \alpha_{\mathcal{A}}\big)\lambda$. The transition probabilities depicted in Fig. 7.6a can be obtained



(a) Mining a mini-PoW



(b) Mining a Bitcoin block

as follows:

$$\alpha'_i = \frac{d\alpha_i}{d(\alpha_C+\alpha_i)+\alpha_{NC}+\alpha_{\mathcal{A}}} \ , \ \alpha'_C = \frac{d\alpha_C}{d(\alpha_C+\alpha_i)+\alpha_{NC}+\alpha_{\mathcal{A}}} \ ,$$
$$\alpha'_{\mathcal{A}} = \frac{\alpha_{\mathcal{A}}}{d(\alpha_C+\alpha_i)+\alpha_{NC}+\alpha_{\mathcal{A}}} \ , \ \alpha'_{NC} = \frac{\alpha_{NC}}{d(\alpha_C+\alpha_i)+\alpha_{NC}+\alpha_{\mathcal{A}}} \ . \tag{7.40}$$

The state probabilities can be calculated as follows:

$$p_0 = \frac{1-\alpha'_{\mathcal{A}}}{1-\alpha'_{\mathcal{A}}+\alpha_{\mathcal{A}}(1+\alpha'_{NC})} \ , \ p_1 = \frac{\alpha_{\mathcal{A}}}{1-\alpha'_{\mathcal{A}}+\alpha_{\mathcal{A}}(1+\alpha'_{NC})} \ ,$$
$$p_2 = \frac{\alpha_{\mathcal{A}}\alpha'_{NC}}{1-\alpha'_{\mathcal{A}}+\alpha_{\mathcal{A}}(1+\alpha'_{NC})} \ . \tag{7.41}$$

The expected return of mining a min-PoW for pool p_i is equal to

$$r_1 = \alpha_i p_0 \lambda R + \alpha_i p_2 (1+\mathtt{br}_3)\lambda R + \alpha'_i p_1 \mathtt{br}_2 \lambda_1 R \ . \tag{7.42}$$

In the second scenario, we consider the case that the petty-compliant pool p_i mines a Bitcoin block. The puzzle mining rate at states s_0 and s_2 are the same as λ. However, the puzzle mining rate at state s_1 is equal to $\lambda_1 = \big(d\alpha_C+\alpha_i+\alpha_{NC}+\alpha_{\mathcal{A}}\big)\lambda$. The transition probabilities depicted in Fig. 7.6b can be obtained as follows:

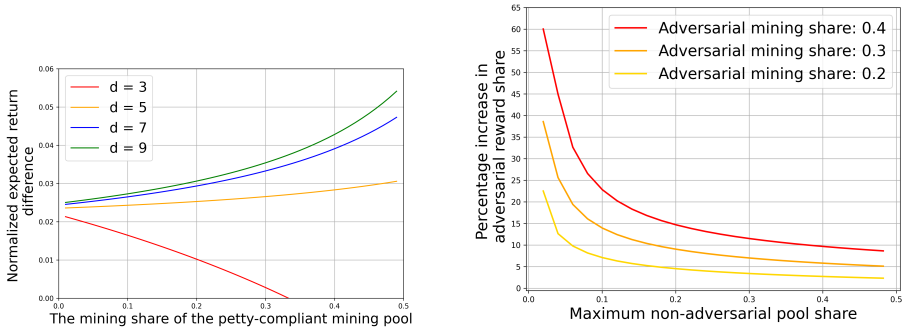
$$\alpha'_i = \frac{\alpha_i}{d\alpha_C+\alpha_i+\alpha_{NC}+\alpha_{\mathcal{A}}} \ , \ \alpha'_C = \frac{d\alpha_C}{d\alpha_C+\alpha_i+\alpha_{NC}+\alpha_{\mathcal{A}}} \ ,$$
$$\alpha'_{\mathcal{A}} = \frac{\alpha_{\mathcal{A}}}{d\alpha_C+\alpha_i+\alpha_{NC}+\alpha_{\mathcal{A}}} \ , \ \alpha'_{NC} = \frac{\alpha_{NC}}{d\alpha_C+\alpha_i+\alpha_{NC}+\alpha_{\mathcal{A}}} \ . \tag{7.43}$$

The state probabilities can be calculated as follows:

$$p_0 = \frac{1-\alpha'_{\mathcal{A}}}{1-\alpha'_{\mathcal{A}}+\alpha_{\mathcal{A}}(1+\alpha'_{NC}+\alpha'_i)} \ , \ p_1 = \frac{\alpha_{\mathcal{A}}}{1-\alpha'_{\mathcal{A}}+\alpha_{\mathcal{A}}(1+\alpha'_{NC}+\alpha'_i)} \ ,$$
$$p_2 = \frac{\alpha_{\mathcal{A}}(\alpha'_{NC}+\alpha'_i)}{1-\alpha'_{\mathcal{A}}+\alpha_{\mathcal{A}}(1+\alpha'_{NC}+\alpha'_i)} \ . \tag{7.44}$$

The expected return of mining a Bitcoin block for pool p_i is equal to

$$r_2 = \alpha_i p_0 \lambda R + \alpha_i p_2 \left(\frac{\alpha'_{NC}}{\alpha'_{NC}+\alpha'_i}\right)(1+\mathtt{br}_3)\lambda R + 2\alpha_i p_2 \left(\frac{\alpha'_i}{\alpha'_{NC}+\alpha'_i}\right)\lambda R \ . \tag{7.45}$$



(a) Mini-PoW mining vs Bitcoin mining. (b) Percentage increase in reward share.

Figure 7.7: Quantitative analysis of the distraction attack.

7.A5.2 Quantitative Analysis of the Distraction Attack

For an adversarial mining power share of $\alpha_{\mathcal{A}} = 0.4$ and the mini-PoW mining reward of $\texttt{br}_2 = 0.1 \times \alpha_{\mathcal{A}}$, Fig. 7.7a represents the normalized expected return difference Δ of a petty-compliant pool based on its mining share, for different values of the difficulty ratio d. As can be seen, for an incentivizing factor of $\epsilon = 0.02$, all petty-compliant mining pools prefer mini-PoW mining over Bitcoin mining if the difficulty ratio is set to $d \geq 5$. Note that, as the mini-PoW mining reward is set to $\frac{1}{10}$ of the adversarial mining share, according to Lemma 7.11, the maximum difficulty ratio that can result in a profitable distraction attack for the adversary is $d = 10$. For the same mini-PoW reward, the smaller the difficulty ratio, the more profitable the distraction attack would be for the adversary. However, setting the difficulty ratio to a low value limits the portion of petty-compliant miners who follow mini-PoW mining.

Figure 7.7b depicts the percentage increase in the adversarial reward share as a function of the mining share of the largest non-adversarial mining pool in the network, for a mini-PoW mining reward of $\texttt{br}_2 = 0.1 \times \alpha_{\mathcal{A}}$ and $\epsilon = 0$. To generate Fig. 7.7b, we calculated the minimum difficulty ratio that makes mini-PoW mining the dominant strategy for all petty-compliant mining pools. Using this difficulty ratio, we then determined the corresponding adversarial reward share. In App. 7.A5.5, we discuss how the distraction attack can be further extended to be applicable using non-adversarial blocks.

7.A5.3 BDoS Attack Description

Let B_0 denote the tip of the canonical chain. In the BDoS attack, when the adversary mines a new block B_1 on top of B_0, it only publishes the header of block B_1. As a result, other miners are unable to mine on top of B_1 since they cannot confirm the correctness of the transactions included in it. This situation places miners in a dilemma: whether to stop mining or continue mining on top of block B_0. Mining on top of B_0 is risky; if they successfully mine another block on top of it, that block will enter a fork race with block B_1, potentially leading to it being orphaned. Under a BDoS attack, a victim pool may decide to stop mining, depending on the mining costs incurred. If the mining reward is significantly higher than the mining cost, the miner will likely continue mining on top of block B_0. The BDoS attack primarily affects mining pools whose mining revenue is only slightly above their mining costs, making it ineffective against miners with a higher profit-to-loss ratio. In contrast, the introduced distraction attack can disincentivize mining on Bitcoin, even when miners assume that there are no mining costs. In our distraction attack, once the adversarial mining pool announces mining of the next block, other mining pools face three possibilities: 1) continue mining on top of the Bitcoin chain, 2) completely stop mining, or 3) engage in mini-PoW mining. The first two possibilities are those covered in the BDoS attack. The third possibility, which is new to the distraction attack, expands the effect of the attack to a broader range of mining pools rather than only those suffering from high mining costs.

Another point regarding the BDoS attack is that the adversarial mining pool always faces a fork race, which can result in the orphaning of its block. However, in our distraction attack, by publishing the block after solving a mini-PoW puzzle, the adversary can avoid the fork race and save its block while still conducting a successful mining power destruction attack.

7.A5.4 Revealing Transactions

To ensure the adversary reveals block transactions, the smart contract must provide a URL to all transactions (excluding the coinbase transaction), making them visible to all pools. The adversarial pool must reveal the coinbase transaction, which includes an extra nonce, upon submitting a valid mini-PoW. The contract should also include parameters to verify that the Merkle root of the transactions and the coinbase transaction matches the stored Merkle root. This smart contract allows petty-compliant mining pools to verify the correctness of block transactions before starting mini-PoW mining.

7.A5.5 Distraction Attack Using the Non-Adversarial Blocks

In Sect. 7.A5, we analyzed the mining power distraction attack conducted by an adversarial mining pool using its own blocks. In the extended attack, the adversarial mining pool incentivizes a petty-compliant mining pool to only publish its block to the adversarial mining pool while hiding it from the other petty-compliant mining pools. The block gets revealed to other petty-compliant mining pools only upon receiving a valid mini-PoW.

7.A5.5.1 Attack Description.

The adversarial mining pool requests a petty-compliant mining pool to only share its block B with the adversarial mining pool. Once the adversarial pool verifies the validity of block B, it deploys the same contract as mentioned in Sect. 7.A5 for block B, where block B is hidden from all petty-compliant mining pools except for its miner. In addition to the 3 bribes discussed in Sect. 7.A5, the adversarial mining pool deposits another normalized bribe $\mathtt{br}_4 = \epsilon$ in the smart contract. The miner of block B is eligible to withdraw $\mathtt{br}_4$ if a user submits a valid mini-PoW.

Note that in this attack, there is no need for the miner of block B and the adversarial mining pool to trust each other as the attack is risk-free for both of them. If the petty-compliant mining pool shares its block only with the adversarial mining pool and observes that the adversarial pool does not deploy the contract, it can promptly publish the block to all other mining pools. Similarly, if the adversarial mining pool deploys the contract for the block but then the block gets published by its miner, the adversarial mining pool does not lose anything as the bribe is payable only upon the submission of a mini-PoW.

Chapter 8

Bitcoin under Volatile Block Rewards: How Mempool Statistics Can Influence Bitcoin Mining

Publication Data

Roozbeh Sarenche, Alireza Aghabagherloo, Svetla Nikova, and Bart Preneel, "Bitcoin under Volatile Block Rewards: How Mempool Statistics Can Influence Bitcoin Mining," to appear in *ACM Conference on Computer and Communications Security (CCS)*, 2025 (https://arxiv.org/abs/2411.11702).

Contributions

I was the main author involved in Conceptualization, Methodology, Writing Original Draft, Software, and Visualization.

Abstract. The security of Bitcoin protocols is deeply dependent on the incentives provided to miners, which come from a combination of block rewards and transaction fees. As Bitcoin experiences more halving events, the protocol reward converges to zero, making transaction fees the primary source of miner rewards. This shift in Bitcoin's incentivization mechanism, which introduces volatility into block rewards, leads to the emergence of new security threats or intensifies existing ones. Previous security analyses of Bitcoin have either considered a fixed block reward model or a highly simplified volatile model, overlooking the complexities of Bitcoin's mempool behavior.

In this paper, we present a reinforcement learning-based tool for analyzing mining strategies under a more realistic volatile reward model. The tool leverages the Asynchronous Advantage Actor-Critic (A3C) algorithm to derive near-optimal strategies while interacting with an environment that simulates the behavior of the Bitcoin mempool during any specified period, enabling analysis based on actual historical patterns. It supports the evaluation of adversarial mining strategies, such as selfish mining and undercutting, both before and after the difficulty adjustment, offering insights into the effects of mining attacks in both the short and long term.

We revisit the Bitcoin security threshold presented in the WeRLman paper [10] and demonstrate that the implicit predictability of valuable transaction arrivals in this model leads to an underestimation of the reported threshold. Additionally, we show that, while adversarial strategies like selfish mining under the fixed reward model incur an initial loss period of at least two weeks, the transition toward a transaction-fee era incentivizes mining pools to abandon honest mining for *immediate* profits. This incentive is expected to become more significant as the protocol reward approaches zero in the future.

8.1 Introduction

As of December 2024, Bitcoin [21] holds a market capitalization of nearly 1.8 trillion dollars [4], surpassing the annual Gross Domestic Product (GDP) of over 94% of countries [5]. While Bitcoin has progressed without major security attacks thus far, its steady growth over the past decade and a half does not guarantee future security, highlighting the need to address potential vulnerabilities. Protecting Bitcoin's economy requires ensuring the resilience of its incentivization mechanism. If consensus nodes find their rewards insufficient to cover costs, they may abandon the chain, risking a halt in blockchain operations. Bitcoin's incentivization comes from two sources: the *protocol reward* for mining a block and the *transaction fee* paid by users. In Bitcoin's early years, the protocol reward significantly outweighed transaction fees. However,

due to the halving mechanism, which reduces the protocol reward every four years, transaction fees are expected to match and eventually replace the protocol reward as the sole source of incentivization. In fact, there have already been periods, such as in December 2023 and April 20, 2024, when transaction fees equaled or even exceeded the protocol reward [2].

As the balance between transaction fees and the protocol reward changes over time, the incentivization dynamics of Bitcoin change as well, potentially leading to the emergence of new security threats or intensifying existing ones. One consequence of transitioning to a transaction-fee-driven era is that the total reward per block can no longer be considered *fixed*. Block rewards become *volatile*, as the inclusion of transactions with varying fee levels causes fluctuations in the total reward per block. This volatility in block rewards alters the profitability of various mining strategies. For example, under the fixed reward model, once a miner receives a new block that extends the longest adopted fork in its view, the miner is incentivized to accept the new block and start mining on top of it [23]. Since block rewards are fixed, there is no advantage in taking the risk of orphaning an already-mined block. However, in a volatile block reward model, a miner might find it more profitable to attempt to orphan a newly mined block with significantly high transaction fees to steal those transactions [10, 12], a strategy referred to as *undercutting*. Additionally, a miner may engage in *selfish mining* by creating a private fork of blocks to take advantage of including all available transactions from the mempool in its private fork. These examples highlight the need to revisit the security analyses conducted under the fixed reward model [15, 23, 28] to better understand Bitcoin's future security in the context of volatile block rewards.

Some research has analyzed Bitcoin's security under the volatile block reward model in a simplified scenario. Carlsten et al. [12] raised concerns about Bitcoin security as the protocol reward diminishes to zero, introducing the undercutting attack and showing how selfish mining could become more threatening in a transaction-fee-driven era. While the paper highlights Bitcoin's instability under the volatile reward model, the analysis relies on simplified assumptions, such as unlimited block space, allowing blocks to collect all available transaction fees. Additionally, the analysis is based on a basic selfish mining strategy, overlooking the optimal strategy and other actions that adversarial miners might use in the volatile block reward model. To improve mining analysis in the transaction-fee setting, the authors in [10] introduced the WeRLman framework, which combines deep reinforcement learning with a Monte Carlo Tree Search (MCTS) tool, inspired by the AlphaGo Zero implementation [26]. They assumed occasional *whale* transactions with significantly high fees that can result in volatility of block rewards. The study showed that the minimum hash rate required for a profitable deviation from honest mining, referred to as the *security threshold*, decreases as Bitcoin undergoes more halvings. While

the WeRLman paper made progress in analyzing Bitcoin security, its mempool implementation and treatment of transaction fees do not accurately reflect real-world conditions. In WeRLman, a block can include either a normal transaction or a whale transaction, resulting in blocks with only two distinct fee levels. This assumption simplifies the complex mempool environment to consider solely the count of whale transactions, all sharing the same fee. Moreover, WeRLman is designed to derive a near-optimal mining strategy only in a setting where the mining difficulty has been adjusted according to the adopted strategy. However, in practice, if a miner deviates from the honest strategy, it takes an epoch of at least two weeks for the difficulty to be adjusted, a relatively long period that WeRLman overlooks in its analysis.

In this paper, we enhance the analysis of mining strategies in Bitcoin by incorporating a more realistic volatile block reward model that obtains the near-optimal strategies both before and after the difficulty adjustment. We begin our analysis by revisiting the Bitcoin security threshold presented in the WeRLman paper [10]. We demonstrate that, due to the design of the WeRLman environment, an adversary in this setting can predict the arrival of the next whale transaction before deciding on its action—an ability we refer to as *predictive capability*. We argue that the predictive capability strengthens the adversary compared to a real-world adversary, resulting in an underestimation of the security threshold. We modify the WeRLman environment to transform it into a non-predictable setting, where the adversary decides its action solely based on the mempool status at the moment of block generation, without any foresight regarding the types of subsequent transactions. The Bitcoin security threshold obtained under the non-predictable WeRLman model, while still lower than that in the fixed block reward model, is higher than what can be achieved under the original WeRLman environment.

We proceed by introducing a simplified model to analyze mining strategies, in which the volatility of block rewards arises from the varying generation times of blocks. We argue that block reward volatility is not solely attributed to the infrequent arrival of whale transactions. The inherent competition among transactions to be included in the chain quickly causes blocks with longer generation times to include higher transaction fees, resulting in block reward volatility even in the absence of whale transactions. To this end, we build our simplified model upon the observed positive correlation between block generation time and its fee. Compared to the WeRLman environment, our simplified environment includes a greater number of blocks with varying fee levels, allowing us to analyze more fine-grained strategies. We employ a Markov Decision Process (MDP)-based tool to analyze the profitability of mining strategies under our simplified volatile model. Our MDP tool estimates a more accurate security threshold for Bitcoin in the transaction-fee era compared to the non-predictable WeRLman model. However, this threshold serves as an

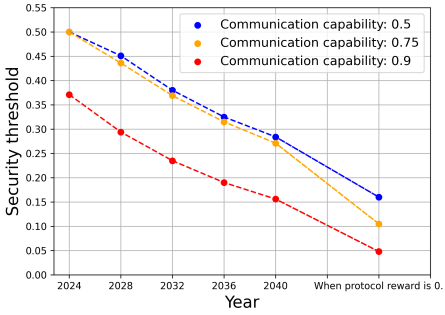


Figure 8.1: Bitcoin security threshold before DAM.

upper bound on the actual value, primarily due to the limited number of states the MDP can handle and the restriction of decision-making to block mining events.

In the next step, we analyze the profitability of mining strategies before difficulty adjustment. An interesting observation under the volatile reward model is that a deviant strategy, such as a selfish mining attack, can become profitable immediately after launch without needing to wait for difficulty adjustment. As discussed in the literature [15, 16, 22, 24], under the fixed reward model assumption, a selfish miner incurs losses during the initial epoch of attack when the mining difficulty has not yet adjusted, the duration of which is at least two weeks. However, under the volatile reward model, this loss period is shortened or may disappear entirely, making selfish mining a more significant threat in the transaction-fee era. Using our MDP-based tool within the simplified model, we analyze the Bitcoin security threshold prior to difficulty adjustment, which has not been explored in the WeRLman paper. Figure 8.1 illustrates the Bitcoin security threshold in the upcoming year as Bitcoin undergoes more halvings, based on different adversarial communication capabilities[1]. Miners with a mining share above this threshold are not only incentivized to deviate from the honest strategy, but also do not face any initial loss. For a typical communication capability of 50%, when the protocol reward approaches 0, the Bitcoin security threshold reduces to 16%, which is within the range of current mining pool shares.

Due to the limited state space in the simplified model and the simplifying assumptions made for implementation, the simplified model calculates a lower bound on mining strategy profitability. To conduct a more precise analysis of

[1]Communication capability is defined as the ratio of honest miners who receive the adversarial block first in the case of a block race.

mining strategies under the volatile reward model, we propose an enhanced model that improves both the implementation of mempool behavior and the training approach. Specifically, we apply the Asynchronous Advantage Actor-Critic (A3C) reinforcement learning algorithm [20] to derive a near-optimal mining strategy. In this algorithm, multiple agents (representing a payoff-maximizing miner) interact with their own instance of the environment (the Bitcoin blockchain) to train a global policy simultaneously. To better model the Bitcoin mempool in the A3C environment, we analyze historical Bitcoin mempool data to estimate the transaction fee arrival patterns. In our implementation, transactions are categorized into different groups based on the fee paid per unit of weight. We estimate how the total transaction weight grows over time for each category, incorporating these estimates into our environment. Our A3C implementation considers a larger set of actions and states compared to models designed for the fixed reward system, enabling it to obtain near-optimal policies both before and after difficulty adjustments under the volatile reward model. The undercut action is modeled as a continuous action in our A3C model, helping the adversary determine when and for how long to undercut the tip of the canonical chain.

To summarize, our key contributions are:

- Revisiting the security threshold of difficulty-adjusted Bitcoin mining from the WeRLman paper [10], based on a real-world adversary without predictive capability (Section 8.3).
- Introducing a simplified volatile reward model that includes a greater number of block reward levels and an MDP-based tool to achieve a more precise security threshold (Section 8.4).
- Analyzing the security threshold of Bitcoin mining under the volatile model before difficulty adjustment to highlight the security threat regarding the immediate profitability of adversarial strategies (Section 8.5).
- Implementing an A3C-based tool to derive a near-optimal mining strategy under a realistic mempool model (Section 8.6).

To present the results in this paper, we have run our implementation using real-world mempool statistics [8] from different periods to demonstrate the threat of adversarial deviation from the honest strategy if the same mempool patterns reoccur.

8.2 Preliminaries

8.2.1 System Model and Definitions

The honest strategy in the Bitcoin protocol is defined as follows:

Definition 8.2.1 (Honest strategy)**.** A mining node should always mine on top of the longest chain [2] available in its view. Once a new block is mined, the mining node should immediately publish the block to all the other mining nodes.

The mining nodes in our scheme are categorized into two groups:

- *Honest* mining nodes: These nodes, which are denoted by $\mathcal{H}$, follow the honest strategy defined in Definition 8.2.1.
- *Adversarial* mining nodes: In our paper, we assume that the *adversary*, denoted by $\mathcal{A}$, controls a subset of mining nodes referred to as adversarial nodes. These adversarial nodes can arbitrarily deviate from the honest strategy. Namely, they can choose any fork rather than the longest chain to mine on top of it or withhold the mined blocks.

Depending on the context, we may refer to adversarial nodes as an attacker, a selfish miner, or an undercutter. In selfish mining, an attacker withholds a newly mined block instead of immediately broadcasting it to the network. In undercutting, the attacker attempts to orphan the tip of the canonical chain, forcing the transactions it contains back into the mempool. A brief discussion of mining attacks that an adversary can conduct in our model, including selfish mining and undercutting, is presented in Appendix 8.A1.

Although the longest chain fork choice rule outputs a deterministic chain in most cases, situations may arise in which there are multiple forks with the same height in the view of a mining node (the same-height fork race), making the chain selection somewhat subjective. Based on how mining nodes select the winning chain in the presence of multiple same-height forks, the honest nodes are divided into 2 sub-groups:

- Altruistic mining nodes: These nodes always choose a fork they have heard about earlier in the case of a same-height fork race.
- Petty-compliant mining nodes: These nodes choose a fork that leads to a greater profit for them in the case of a same-height fork race.

In a same-height fork race, petty-compliant nodes choose the fork with the lower total transaction fees included. The rationale behind this choice is that the fork with fewer transaction fees leaves more fees in the transaction pool, allowing the next block mined on this fork to include a higher amount of fees. Assuming all mining nodes share the same view of the transaction pool, in a same-height fork race, the winning chain for a petty-compliant node is the one whose tip block (the highest block) was mined earlier and, as a result, includes lower fees.

[2] In Proof-of-Work blockchains, the longest chain is a chain with the highest accumulated difficulty.

Another factor, besides mining share, that can affect the adversary's profitability in following a deviant strategy instead of honest mining is its connectivity to other mining nodes. We define adversarial *communication capability* as follows:

Definition 8.2.2 (Adversarial communication capability)**.** The adversarial communication capability is defined as the ratio of altruistic nodes that receive the adversarial fork earlier than the honest fork in the case of a same-height fork race.

The adversarial communication capability specifies the altruistic node ratio that selects the adversarial chain as the winning one in a same-height fork race.

In our model, time is assumed to be divided into rounds, denoted by r. In each round, a mining node can calculate multiple mining (hash) queries, the number of which is proportional to its mining share. A time step, denoted by t, represents the set of consecutive rounds after which a block is mined in the system. At each time step t, a block is generated by either the adversary or the honest miners. However, depending on the adversary's strategy, the block generated at time step t might not be immediately added to the canonical chain. Instead, it could be either added to the canonical chain or eventually identified as an orphan block (excluded from the canonical chain) at a future time step.

Definition 8.2.3 (Block ratio)**.** The block ratio of adversary $\mathcal{A}$ following strategy π is defined as follows:

$$\begin{aligned} \mathtt{BR}_T^{\mathcal{A}}(\pi) &= \frac{\sum_{t=0}^{T} N_{\mathcal{A}}(t;\pi)}{\sum_{t=0}^{T} N_{\mathcal{A}}(t;\pi) + N_{\mathcal{H}}(t;\pi)}, \quad \text{and} \\ \mathtt{BR}^{\mathcal{A}}(\pi) &= \lim_{T\to\infty} \mathtt{BR}_T^{\mathcal{A}}(\pi) \ , \end{aligned} \tag{8.1}$$

where $N_{\mathcal{A}}(t;\pi)$ and $N_{\mathcal{H}}(t;\pi)$ denote the number of adversarial blocks and the number of honest blocks, respectively, added to the canonical chain at time step t under policy π.

Definition 8.2.4 (Time-averaged profit)**.** The time-averaged profit (per-round profit) of adversary $\mathcal{A}$ following strategy π is defined as follows:

$$\begin{aligned} \mathtt{Profit}_T^{\mathcal{A}}(\pi) &= \frac{\sum_{t=1}^{T} R_{\mathcal{A}}(t;\pi) - C_{\mathcal{A}}(t;\pi)}{T}, \quad \text{and} \\ \mathtt{Profit}^{\mathcal{A}}(\pi) &= \lim_{T\to\infty} \mathtt{Profit}_T^{\mathcal{A}}(\pi) \ , \end{aligned} \tag{8.2}$$

where $R_{\mathcal{A}}(t;\pi)$ and $C_{\mathcal{A}}(t;\pi)$ denote the revenue achieved and the mining cost incurred by adversary $\mathcal{A}$ at time step t under policy π, respectively.

The revenue a miner receives at a given time step equals the total rewards from its blocks added to the canonical chain at that time. Each block reward consists of the protocol reward and the transaction fees included in the block.

Definition 8.2.5 (Security threshold)**.** For a given security parameter ϵ, the security threshold of a mining strategy π is defined as the minimum mining share required for the time-averaged profit of strategy π to exceed the time-averaged profit of the honest strategy by at least ϵ. The security threshold within an environment is defined as the minimum mining share for which the profitability of the corresponding optimal strategy exceeds the time-averaged profit of the honest strategy by at least ϵ.

8.2.2 Overview of Selfish Mining Profitability

In Bitcoin, a difficulty epoch is defined as the period during which 2016 blocks are appended to the canonical chain. At the end of each epoch, the Difficulty Adjustment Mechanism (DAM) recalibrates the difficulty target for the subsequent epoch based on the hash power estimated from the previous epoch. Adversarial behaviors that destroy a portion of the network's mining power can cause DAM to underestimate the active mining power. This underestimation lowers mining difficulty, potentially making such behaviors profitable. In the following, we briefly review the well-established fact that, under the fixed reward model, adversarial behaviors like selfish mining are unprofitable before difficulty adjustment [15, 22, 25]. However, they can become profitable over the long term after applying DAM.

8.2.2.1 Non-profitability of Pre-DAM Selfish Mining under the Fixed Block Reward Model

Under a fixed block reward model, selfish mining cannot be more profitable than honest mining during the first epoch of the attack and before the DAM is applied [15]. To compare profits from honest and selfish mining before difficulty adjustment, let e denote an epoch, λ the mining rate, α the mining share of adversary $\mathcal{A}$, and R the fixed reward per block. We define the adversary's *total* and *canonical* block generation rates as the average number of rounds in which a new adversarial block is mined, and the average number of rounds in which a new adversarial block is added to the canonical chain, respectively. Under honest mining, the adversary's canonical block generation rate is $\lambda\alpha$, yielding a time-averaged profit of $\lambda\alpha R$. If the adversary $\mathcal{A}$ starts selfish mining at the beginning of epoch e, its canonical block generation rate would be less than or equal to that under honest mining ($\leq \lambda\alpha$). This is because the difficulty of epoch e is already set before the start of the epoch, and the adversary's

strategy during the epoch cannot influence the epoch difficulty. Since both the epoch difficulty and the adversary's mining share are fixed, the adversary's total block generation rate remains the same under both honest and selfish mining. However, under selfish mining, some of the adversarial blocks may become orphaned, leading to a decrease in the adversary's canonical block generation rate. Therefore, the time-averaged profit from selfish mining during epoch e (before a DAM is applied) is $\leq \lambda\alpha R$. Intuitively, before a DAM is applied, the adversary cannot mine blocks faster, regardless of the strategy chosen [25].

Note that while selfish mining may potentially increase the adversary's block ratio during epoch e, this increase is due to orphaned honest blocks and does not reflect an increase in the adversary's time-averaged profit.

8.2.2.2 Profitability of Long-Range Selfish Mining under the Fixed Block Reward Model

The adversary needs to sustain the selfish mining attack for a longer period to achieve profitability. In Bitcoin, DAM is responsible for adjusting the block generation rate to ensure that it takes, on average, 10 minutes to mine a new block. This implies that the ideal epoch duration is two weeks. Assume the adversary starts selfish mining at the beginning of epoch e. As a result of selfish mining, some of the blocks (both adversarial and honest) mined during epoch e may become orphaned. The generation of these orphan blocks extends the duration of epoch e because to add 2016 blocks to the canonical chain, more than 2016 blocks need to be mined. At the end of epoch e, which exceeds two weeks, the DAM decreases the mining difficulty for the following epoch to restore the canonical block generation rate to one block per 10 minutes. This reduction in difficulty leads to a higher block generation rate for the adversary, indicating that starting from epoch $e + 1$, the adversary can mine new blocks in fewer rounds and consequently increase its time-averaged profit [24]. This implies that under the fixed block reward model, selfish mining requires at least one difficulty epoch to become profitable. In practice, the attack duration must be extended further to ensure profitability. This is because the selfish miner incurs losses during the initial epoch due to orphaned blocks, necessitating a longer attack period to compensate for these losses.

8.2.3 Asynchronous Advantage Actor-Critic (A3C)

Reinforcement learning (RL), a branch of artificial intelligence (AI), aims to develop autonomous agents that learn optimal policies to maximize long-term rewards through iterative interactions with their environment [9]. Asynchronous Advantage Actor-Critic (A3C) [20] is a policy gradient method within the RL

domain that simultaneously optimizes both the policy and value function estimation. The policy, denoted by π, guides the agent's actions in the environment by assigning probabilities to each possible action in a given state, indicating the likelihood of the agent choosing that particular action. The value function at a given state s, denoted by $V(s;\pi)$, estimates the expected cumulative reward that an agent can obtain starting from state s and following a particular policy π thereafter, formally defined as follows:

$$V(s;\pi) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s, \pi\right] \ , \tag{8.3}$$

where s_t is the state visited at step t, r_t is the reward agent receives at step t, and γ is the discount factor. A3C uses multiple agents that interact with their own instances of the environment in parallel. These agents run independently and asynchronously to explore different parts of the state space simultaneously, reducing the correlation between experiences and improving exploration. Unlike many reinforcement learning algorithms, such as deep Q-learning (DQN), which rely on experience replay (storing and reusing past experiences), A3C updates the network in real-time as experiences are generated. The diversity and independence of experiences gathered by multiple agents compensate for the lack of experience replay. A3C uses 1-to-n-step returns to update both the policy and the value function. These updates occur after every T steps or when a terminal state is reached. The value loss for the trajectory over the time steps $t \in \{0, 1, \cdots, T-1\}$, denoted by L_v, is defined as the Mean Squared Error (MSE) between the value function and the expected return as follows:

$$L_v(\theta_v) = \frac{1}{2}\mathbb{E}_t\Big[\big(R_t - V(s_t;\theta_v)\big)^2\Big] \ , \tag{8.4}$$

where R_t is the expected return in step t with respect to the updating time step T and is defined as follows:

$$R_t = \sum_{k=0}^{T-1-t} \gamma^k r_{t+k} + \gamma^{T-t} V(s_T;\theta_v) \ . \tag{8.5}$$

The main goal of the policy update in A3C is to improve the policy π such that actions with higher advantages are more likely to be chosen. The policy loss function for the trajectory over the time steps $t \in \{0, 1, \cdots, T-1\}$, denoted by L_π is formulated as follows:

$$L_\pi(\theta_\pi) = \mathbb{E}_t\Big[\log \pi(a_t|s_t) \cdot A_t\Big] \ , \tag{8.6}$$

where $\pi(a_t|s_t)$ represents the probability of taking action a_t at state s_t according to the current policy, and A_t is the advantage function defined as $R_t - V(s_t)$.

Note that while the policy parameters (θ_π) and value function parameters (θ_v) are typically treated as separate for generality, in practice, some of these parameters are often shared.

8.3 Revisiting the WeRLman Model

The WeRLman paper [10] studied selfish mining under the volatile block reward model, assuming that mining difficulty is well-adjusted during the attack. The WeRLman authors showed that the security threshold of Bitcoin mining, defined in Definition 8.2.5, is lower in the volatile reward model compared to the fixed block reward model. This is because, in the volatile block reward model, some blocks may occasionally be mined that include significantly higher rewards due to the inclusion of valuable transaction fees. Once such a block is mined, the adversary is incentivized to orphan it rather than extend it, in an attempt to steal the valuable transactions contained within. Although transitions to the volatile reward model can reduce the mining power threshold for profitable selfish mining, we argue that, due to the WeRLman environment design, it underestimates the mining power threshold.

The WeRLman environment. In the WeRLman environment, two types of transactions exist in the mempool: normal transactions with a normalized fee of 1 and whale transactions with a normalized fee of $1 + F$. The model assumes that, at each event of mining a block B with a new height, a single transaction is added to the mempool. This transaction implicitly represents the transactions that arrive in the mempool between the mining time of block B and the next block. More specifically, once a block is mined, the WeRLman environment samples a new transaction and adds it to the mempool. This new transaction is either normal, with probability $1 - p$, or a whale, with probability p. This implementation assumption in the WeRLman model ensures a constant transaction arrival rate but introduces a significant bottleneck that benefits the adversary. In the WeRLman model, at each decision-making point, the adversary knows whether the next transaction added to the mempool will be a whale or a normal transaction before deciding its next action. We refer to this property of knowing the type of the next transaction in advance as *predictive capability*.

Our analysis shows that the implicit predictive capability in the WeRLman model allows the adversary to make more profitable decisions regarding whether to publish or withhold its blocks, thereby increasing its profitability compared to a real-world miner. Note that, in reality, miners do not have knowledge of future transactions in advance. While mining a new block, miners frequently update the contents of their blocks to include the highest-fee transactions

available. Therefore, once a block is mined, it includes the latest available whale transactions, and the miner is unaware of the arrival time of the next whale transaction. To determine Bitcoin's security threshold in the presence of real-world adversaries, we modified the transaction sampling method in the WeRLman environment to eliminate the effect of predictive capability. We refer to this modified version of the WeRLman environment as the *non-predictable* WeRLman, which is explained below.

The non-predictable WeRLman environment. The only difference between the non-predictable WeRLman environment and the original version is that, in the non-predictable environment, the mempool includes only the transactions that arrived up to the block generation event, excluding future transactions. At each event of mining a block B with a new height, the non-predictable environment samples a single transaction that resembles those transactions that arrived between the generation of B's parent block and the generation of B, allowing B to include the transaction. The implementation details of the non-predictable WeRLman environment are provided in Appendix 8.A2.

The authors of the WeRLman paper implemented an MDP tool to analyze the security threshold under the WeRLman environment. For a given adversary with specific mining power, this tool determines the optimal strategy for the adversary within the WeRLman environment. The security threshold is then identified as the minimum mining power at which the optimal strategy deviates from the honest strategy. This MDP tool can also be adapted to derive the optimal strategy for an adversary operating under the non-predictable WeRLman environment. In Figure 8.2, we compare the security threshold for a profitable deviation of honest mining in the WeRLman environment and the non-predictable WeRLman environment. To depict Figure 8.2, we use the same extra fee values F as estimated in the WeRLman paper[3]. Note that as Bitcoin experiences more halving events, the protocol reward decreases, causing the ratio of a whale-included block reward to a normal block reward, i.e., $1 + F$, to increase.

As can be seen in Figure 8.2, the security threshold in the non-predictable environment is higher than in the original WeRLman environment.

[3]In the WeRLman paper, the authors analyzed Bitcoin blockchain data to estimate the extra fee value F for a whale transaction, assuming whale transactions arrive with a frequency of $p = 0.001$. Based on the current protocol reward of 3.125 BTC, and the protocol rewards after 1, 2, 3, 4, and 7 halvings, the estimated values of F are 0.26, 0.45, 0.74, 1.14, 1.58, and 3.2, respectively.

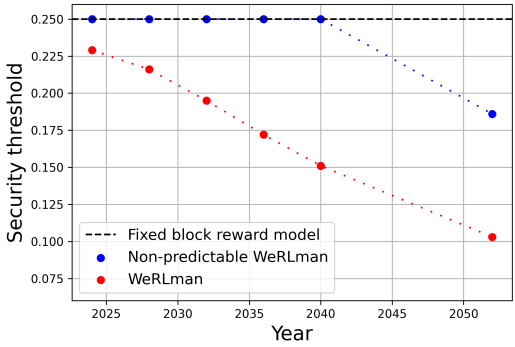


Figure 8.2: Bitcoin security threshold in upcoming years.

8.3.1 Rationale for Higher Security Threshold in Non-Predictable WeRLman

To explain the reason behind the higher security threshold in the non-predictable environment, we define three adversarial strategies: π_1^{WeRLman}, $\pi_1^{\text{Non-predictable}}$, and $\pi_2^{\text{Non-predictable}}$. The first strategy applies only in the original WeRLman environment, as it leverages predictive capability, while the other two are suited for the non-predictable WeRLman environment, where no predictive capability exists. We analyze the profitability of these strategies to identify the factors limiting the adversary's profitability in the non-predictable setting. These three strategies are precisely defined in the Appendix 8.A3, and the main idea behind these strategist are defined as follows:

Strategy π_1^{WeRLman}. This strategy applies exclusively to the WeRLman environment. Once the adversary mines a normal block and knows that the next transaction added to the mempool is a whale transaction, it withholds the block and creates a private fork. [4]. In other scenarios, the adversary follows honest behavior.

Strategy $\pi_1^{\text{Non-predictable}}$. This strategy is suitable for a non-predictable WeRLman environment. Once the adversary mines a normal block, it withholds the block and creates a private fork. In other scenarios, the adversary follows honest behavior.

Strategy $\pi_2^{\text{Non-predictable}}$. This strategy is suitable for a non-predictable WeRLman environment. Once the honest miners mine a whale-included block,

[4]By following this strategy, the adversary creates a private fork that increases its chances of winning the whale transaction available in the mempool.

the adversary does not accept the block but continues mining on the parent of the published block to undercut it. In other scenarios, the adversary follows honest behavior.

In the Appendix 8.A3, we theoretically analyze the three introduced strategies and determine their profitability. In Figure 8.3, we compare the security threshold under the original WeRLman and non-predictable WeRLman environments with the security thresholds of the introduced strategies: π_1^{WeRLman}, $\pi_1^{\text{Non-predictable}}$, and $\pi_2^{\text{Non-predictable}}$. This comparison is based on different values of F, a whale transaction frequency of $p = 0.001$, and a security parameter $\epsilon = 10^{-6}$. The security thresholds for the original WeRLman and non-predictable WeRLman environments are obtained using the MDP tool from the WeRLman paper, while the security thresholds for the introduced strategies are derived from the theoretical analysis presented in Section 8.A3. The comparison shows that the security threshold in the original WeRLman environment overlaps with the security threshold of strategy π_1^{WeRLman}, while the security threshold in the non-predictable WeRLman environment overlaps with the minimum of the security thresholds of $\pi_1^{\text{Non-predictable}}$ and $\pi_2^{\text{Non-predictable}}$. We can conclude that the strategy π_1^{WeRLman} is the deviating strategy that determines the security threshold in the original WeRLman environment. This is because π_1^{WeRLman} is a near-optimal strategy for miners whose mining share is close to the security threshold in the original WeRLman model. However, since strategy π_1^{WeRLman} is not applicable to the non-predictable environment, the security threshold in the non-predictable environment cannot be as low as that in the original WeRLman environment.

Consider the scenario in which the adversary has mined a normal block. Under the WeRLman environment, the adversary has access to additional information

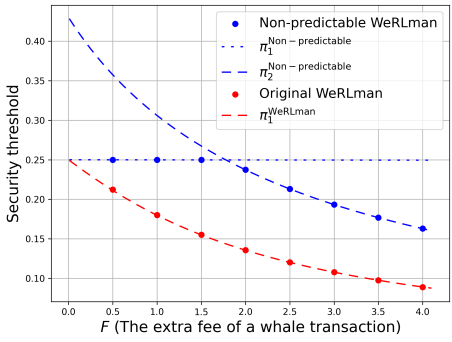


Figure 8.3: Security threshold of adversarial strategies

about the next transaction type and can choose to withhold its normal block only in scenarios where the next transaction is known to be a whale (strategy $\pi_1^{\text{Non-predictable}}$). If the honest miners mine the next block and include the whale transaction, the withheld adversarial block can be used to orphan the honest block and revive the whale transaction. In other words, the adversary under the WeRLman model can withhold its normal block only if it knows that winning a fork race will guarantee the inclusion of the valuable whale transaction. However, under the non-predictable environment, if an adversary with limited mining power mines a new block, it would typically publish the block immediately to secure the reward, as it has no information about the next transaction type. This means that for adversaries with limited mining power, following strategy $\pi_1^{\text{Non-predictable}}$ is not profitable under the realistic non-predictable environment.

8.4 Simplified Volatile Reward Model

As can be seen in Figure 8.2, the non-predictable WeRLman model, despite showing a reduction in the security threshold in the long term (after 2040), cannot precisely assess Bitcoin's threshold in the current and near-future years. This imprecision occurs because the WeRLman environment analyzes adversarial strategies by considering only two types of transactions—normal and whale transactions—while in practice, transactions can have varying fee levels.

In this section, we introduce a new volatile reward model, referred to as the simplified environment. This environment enhances the WeRLman environment by including a greater number of transaction types, each with different fee levels, to provide a more accurate simulation of the real-world mempool. Besides, we present a Markov Decision Process (MDP)-based tool that calculates an optimal strategy under the simplified environment. Our MDP tool is capable of obtaining an *upper bound* for Bitcoin's security threshold in the transaction-fee era. The looseness of this upper bound is due to several reasons. First, the MDP tool can handle only a limited number of states, requiring simplifications in the mempool model, such as assuming a linear relationship between the block's transaction fee and block generation time. Second, in the designed MDP tool, the adversary's decision-making is restricted to block mining events. This means the adversary cannot adjust its strategy between two block mining events based on updates to the mempool status, such as the arrival of a fee-valuable transaction.

8.4.1 Block Generation Time Variability and Reward Volatility

In the WeRLman model, the volatility of block rewards arises from the inclusion of whale transactions. However, it is important to emphasize that the primary source of volatility in block rewards during the transaction fee era is not the occasional arrival of whale transactions that offer significantly higher fees compared to others. Instead, the main factor driving this volatility is the competition among transactions to be included in a block as quickly as possible. This competition forces transactions to adjust their fees based on the state of the mempool. Over time, as more valuable transactions accumulate in the mempool, new transactions need to offer higher fees to have a chance of inclusion in the next block. Therefore, during periods of high demand, block rewards can rise significantly due to the presence of higher-fee transactions in the mempool.

An important factor that increases transaction inclusion demand is extended periods during which no blocks are generated. Due to the random nature of Bitcoin's mining process, block generation times can vary widely, ranging from a few seconds to over an hour. When block generation times are longer, miners have more opportunities to include higher-fee transactions in their blocks, thus increasing block rewards. This phenomenon is evident in the Bitcoin blockchain, where blocks with significantly higher transaction fee rewards compared to their surrounding blocks often correspond to those with unusually long block generation times. In our simplified model, instead of focusing on the concept of whale transactions, we concentrate on block generation time to analyze Bitcoin mining strategies.

8.4.2 Simplified Mempool Environment

In a volatile reward model, miners' profits depend not only on the number of blocks they mine but also on the transaction fees included in those blocks. To reflect the effect of transaction fees on the block rewards, our simplified environment encompasses an equation that calculates the transaction fee included in a block as a function of block generation time. We refer to this equation as the *time-fee* equation. In the simplified mempool environment, we adopt the regression technique introduced in [27] to model mempool behavior using a *linear* time-fee equation. This equation is expressed as $\texttt{fee}(t) = \texttt{fee}_0 + r_{\texttt{fee}}t$, where $\texttt{fee}_0$ and $r_{\texttt{fee}}$ denote the *base fee* and the *fee increase rate*, respectively. The base fee, corresponding to the intercept of the linear equation, represents the average amount of transaction fees remaining in the pool immediately after a block is mined. The fee increase rate, corresponding to the slope, captures the rate at which the collected transaction fees of a block increase over time. Since the simplified model uses an MDP-based tool to analyze selfish mining profitability, it is necessary to simplify the time-fee

equation to fit within a limited number of states. To this end, we divide the linear time-fee equation into $M \geq 2$ discrete time steps. A detailed discussion of the discrete linear time-fee equation used in the simplified model is provided in Appendix 8.A4.

8.4.3 MDP-Based Tool

In this section, we introduce an MDP-based tool to analyze the profitability of selfish mining under the simplified environment introduced in Section 8.4.2. Our MDP-based tool, presented in [7], is designed to calculate a lower bound for selfish mining profitability under the volatile reward model, enabling us to determine the security threshold both before and after the difficulty adjustment.

In our simplified model, we consider two forks: an honest fork and an adversarial fork. The honest fork is publicly known to all miners, while the adversarial fork is known only to the selfish miner, who can strategically decide when to publish the blocks in the adversarial fork. The actions available to the adversary are similar to those introduced in [23] and include: `override`, `adopt`, `match`, and `wait`. Each state in our simplified environment is represented by five elements: $(l_{\mathcal{A}}, l_{\mathcal{H}}, T_{\texttt{total}}, T_{\texttt{last}}, \texttt{fork})$. Here, $l_{\mathcal{A}}$ and $l_{\mathcal{H}}$ denote the lengths of the adversarial and honest forks, respectively, and can range from 0 to `maxForkLen`, where `maxForkLen` represents the maximum fork length in this environment. If one of the forks reaches `maxForkLen`, the selfish miner must either publish its private fork or adopt the honest fork. $T_{\texttt{total}}$ represents the total sum of block generation time steps for all the blocks in the adversarial fork. Given the assumption of a linear time-fee equation, the reward of the adversarial fork can be calculated using $T_{\texttt{total}}$ without the need to store the block generation time for each block separately, as follows: $l_{\mathcal{A}}(R + \texttt{fee}_0) + r_{\texttt{rate}}\Delta T_{\texttt{total}}$, where R is the protocol reward per block. $T_{\texttt{last}}$ represents the time steps elapsed since the mining of the last adversarial block and is used to update the value of $T_{\texttt{total}}$ at each event of adversarial block generation. `fork` can take values 0, 1, or 2, indicating that the latest mined block is adversarial, the latest block is honest, or the action `match` is currently in progress, respectively.

8.4.4 Difficulty-Adjusted Selfish Mining in the Simplified Environment

We begin by discussing the objective function that the MDP tool aims to maximize. Let $\mathbf{t}^{B}_{\texttt{ideal}}$ denote Bitcoin's ideal average block generation time[5]. At each time step in the simplified environment, a new block (either

[5] $\mathbf{t}^{B}_{\texttt{ideal}} = 10$ minutes.

honest or adversarial) is mined, with its block generation time following an exponential process. Let $N_{\mathcal{A}}(t;\pi)$, $N_{\mathcal{H}}(t;\pi)$, and $R_{\mathcal{A}}(t;\pi)$ represent the number of adversarial blocks, honest blocks, and the total reward included in adversarial blocks, respectively, added to the canonical chain at time step t under strategy π. In difficulty-adjusted selfish mining, the average canonical block generation time remains $\mathbf{t}^B_{\texttt{ideal}}$. Thus, the expected elapsed time up to step T is $\mathbf{t}^B_{\texttt{ideal}} \sum_{t=1}^{T} (N_{\mathcal{A}}(t;\pi) + N_{\mathcal{H}}(t;\pi))$. By normalizing $\mathbf{t}^B_{\texttt{ideal}}$ to 1 and ignoring mining costs[6], the time-averaged profit of selfish mining after difficulty adjustment, i.e., the objective function, is given by:

$$\texttt{Profit}^{\mathcal{A}}(\pi) = \lim_{T\to\infty} \frac{\sum_{t=1}^{T} R_{\mathcal{A}}(t;\pi)}{\sum_{t=1}^{T} N_{\mathcal{A}}(t;\pi) + N_{\mathcal{H}}(t;\pi)} \ . \tag{8.7}$$

Note that although the average canonical block generation time under difficulty-adjusted selfish mining remains equal to $\mathbf{t}^B_{\texttt{ideal}}$, the average total block generation time (considering both orphan and canonical blocks) is affected by the adversary's strategy. Therefore, we cannot assume that the total block generation time follows the initial rate λ. Ensuring the correct block generation time is crucial, as it impacts the transaction fee reward. To maintain a constant transaction arrival rate, we adopt the same heuristic as the WeRLman paper. At each step of mining a new block B, we sample an exponential random variable with mean λ to determine the block generation time of B, only if B is a block with a new height[7] in the system. However, if there already exists a block in the opposite fork with the same height as B, the generation time is set to 0. This approach ensures the transaction arrival rate remains consistent regardless of the adversary's strategy.

Security threshold (comparison with WeRLman model). We provide a comparison between the security threshold in our simplified environment and the non-predictable WeRLman environment. The reason our results should not be compared with the security threshold of the original WeRLman environment is that, unlike the original environment, our simplified environment does not grant the adversary predictive capabilities. These predictive capabilities could be incorporated into our model by allowing the adversary to know the block generation time of the next block; however, this would not reflect real-world practices.

To ensure a fair comparison, we adjust the parameters in our model so that they align with the WeRLman model. Let F and $p_{\texttt{whale}}$ denote the additional fee value of a whale transaction and the probability of a whale transaction in the WeRLman model, respectively. Also, let M denote the number of discrete time points in our model, t_{M-1} the greatest time point in our model, λ the

[6] Mining costs remain unchanged unless the adversary turns off some miners.
[7] The block height is the distance of the block from the genesis block.

block generation rate, and $\texttt{fee}(t)$ the time-fee equation. We set t_{M-1} such that the probability of the block generation time being equal to or greater than t_{M-1} matches $p_{\texttt{whale}}$. Specifically, $t_{M-1} = \frac{-\texttt{Ln}(p_{\texttt{whale}})}{\lambda}$. This ensures that the probability of mining a block with the highest possible block generation time (the block with the highest fee) in our model matches the probability of mining a whale-included block in the WeRLman model. Additionally, we configure the time-fee equation $\texttt{fee}(t)$ to satisfy $\frac{\texttt{fee}(t_{M-1})}{\texttt{fee}(t_{\texttt{average}})} = 1 + F$, where $t_{\texttt{average}}$ is the average block generation time according to the discrete time division in our model. This ensures that, in both models, the ratio of the fee of the most valuable block to the fee of a normal block is equal to $1 + F$.

By setting the number of discrete time points $M = 2$, our model exactly matches the non-predictable WeRLman model, as both models only include blocks with two distinct fee levels. Increasing the number of discrete time points allows our model to represent a wider variety of blocks with different rewards, enabling a more precise analysis of profitability in the volatile block reward setting. In Figure 8.4, we compare the security thresholds in the non-predictable WeRLman model and our simplified model, assuming a protocol reward of 0 and a probability of mining the most valuable block of $p = 0.001$. To obtain the security threshold for our model, we used the MDP-based tool introduced in Section 8.4.3, with the number of time points set to $M = 30$. As shown in Figure 8.4, considering an environment with higher block reward volatility decreases the security threshold. This occurs because, in a more volatile model, blocks with rewards lower than a normal 10-minute block exist, allowing the adversary to take the risk of withholding these blocks to orphan valuable honest blocks. Additionally, a greater number of blocks with rewards exceeding the normal reward exist, resulting in more frequent attractive opportunities to deviate from the honest strategy.

8.5 Pre-DAM Selfish Mining Profitability under Volatile Reward Model

In this section, we demonstrate that selfish mining before difficulty adjustment can be profitable under the volatile reward model, enabling miners to deviate from the honest strategy for immediate gain. We first discuss the intuition of how selfish mining, which is unprofitable under the fixed reward model, becomes immediately profitable and thus more threatening in the volatile reward model.

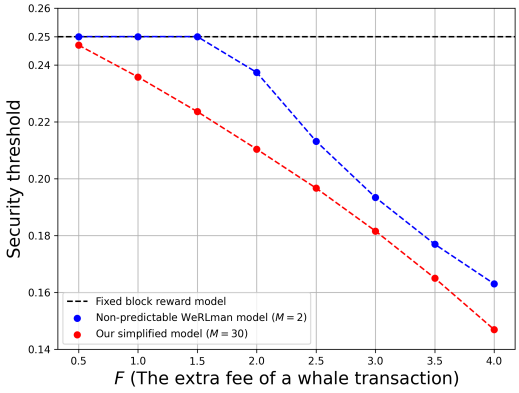


Figure 8.4: Security threshold comparison under our simplified model and non-predictable WeRLman model.

8.5.1 Selfish Mining Profit Lag under the Fixed Block Reward Model

Since Bitcoin's introduction, it has progressed without experiencing long-range or detectable selfish mining attacks. We argue that, in addition to the high hash power required for a successful attack, another contributing factor is the prolonged time required for selfish mining to become profitable under the fixed block reward model. As discussed in Section 8.2.2.1, selfish mining during the first difficulty epoch incurs financial losses for the adversary and must be sustained for several days to become profitable. Figure 8.5 shows the minimum number of days required for a selfish mining attack to exceed the profitability of honest mining under the fixed block reward model, a period referred to as the profit lag in [16]. The figure considers adversaries with communication capabilities of $\gamma = 0.5$, 0.75, and 1 who follow the optimal selfish mining strategy from [23], which maximizes block ratio. The main takeaway is that, under the fixed block reward model, selfish mining demands a significant time investment (more than one ideal epoch or two weeks) to become profitable.

8.5.2 Profitability of Pre-DAM Selfish Mining under the Volatile Block Reward Model

Under the volatile block reward model, selfish mining can yield immediate profits by shortening or eliminating the initial loss period, making it a more

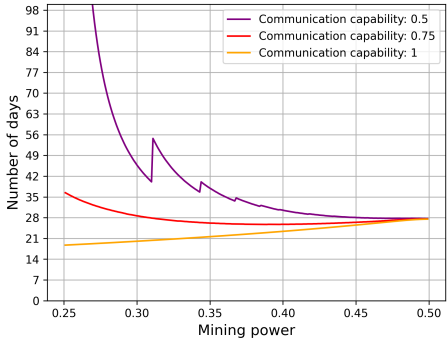


Figure 8.5: The minimum number of days required for a selfish mining attack to become profitable under the fixed reward model.

significant threat to the Bitcoin network. The primary reason for the immediate profitability of selfish mining under the volatile model is that the attack increases the average generation time of adversarial blocks before the first DAM. While this does not affect the protocol reward per block, it boosts the average transaction fee reward per adversarial block.

Let λ_1[8] and R_1 denote the adversarial canonical block mining rate and the average block reward under the volatile reward model when all the miners, including the adversary, mine honestly. Also, let λ_2 and R_2 denote the adversarial canonical block mining rate and the average block reward under the volatile reward model during the first epoch of selfish mining. As discussed in Section 8.2.2.1, the adversary's time-averaged profit during the honest mining and the first epoch of selfish mining are equal to $\lambda_1 R_1$ and $\lambda_2 R_2$, respectively. By conducting a selfish mining attack, the adversarial canonical block generation rate decreases in the first epoch of the attack as some of the adversarial blocks may become orphaned, implying that $\lambda_2 \leq \lambda_1$. However, unlike the fixed block reward model, R_2 is not necessarily equal to R_1. In the volatile model, as a result of the increase in the block generation time[9] during the first epoch of the attack, the average block reward may increase, implying that $R_2 \geq R_1$. The reason behind this increase in the block reward is intuitively illustrated in Figure 8.6. Therefore, one of the time-averaged profits $\lambda_i R_i$ for $i \in \{1, 2\}$ may be greater than the other depending on the adversary's mining parameters and

[8]Note that $\lambda_1 = \lambda\alpha$, where λ is the total block generation rate.

[9]The network block generation time is inversely proportional to the total block generation rate.

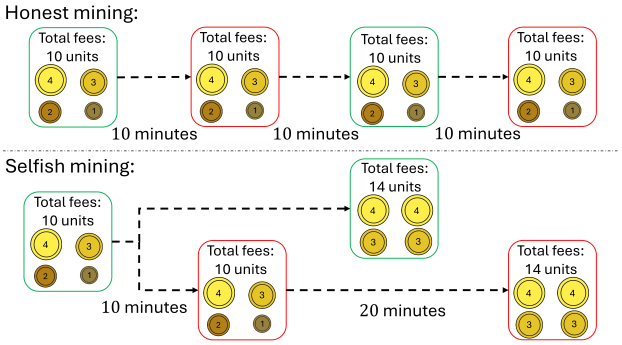


Figure 8.6: Selfish mining under the volatile reward model. Assume that transaction fees can take values $i \in \{1, 2, 3, 4\}$. The rate of transaction arrival for transactions with a fee value $i \in \{1, 2, 3, 4\}$ is one transaction per 10 minutes. Furthermore, each block can include up to 4 transactions. Adversarial (honest) blocks are depicted in red (green). Under honest mining, the block generation time for all blocks is 10 minutes, and each adversarial block receives 10 units of fee. In the case of selfish mining, the block generation time for the second adversarial block increases. If the attack successfully orphans the middle honest block, the second adversarial block can collect an increased fee of 14 units.

the mempool statistics. This shows that under the volatile reward block model, selfish mining can be profitable even in the first epoch of the attack.

8.5.3 Pre-DAM Selfish Mining in the Simplified Environment

In this section, we analyze the profitability of selfish mining before the difficulty adjustment under the simplified model introduced in Section 8.4 using our MDP-based tool. Let $\mathbf{t}^B_{\texttt{ideal}}$ denote Bitcoin's ideal average block generation time, and $R_{\mathcal{A}}(t;\pi)$ represent the total reward included in adversarial blocks added to the canonical chain at time step t under strategy π. Under pre-DAM selfish mining, the average block generation time (considering both orphan and canonical blocks) remains $\mathbf{t}^B_{\texttt{ideal}}$. Since a new block is mined at each step in the environment, the expected elapsed time up to step T is $T\mathbf{t}^B_{\texttt{ideal}}$. By normalizing $\mathbf{t}^B_{\texttt{ideal}}$ to 1 and ignoring mining costs, the time-averaged profit of selfish mining before difficulty adjustment, i.e., the objective function, can be expressed as:

$$\texttt{Profit}^{\mathcal{A}}(\pi) = \lim_{T\to\infty} \frac{\sum_{t=1}^{T} R_{\mathcal{A}}(t;\pi)}{T} \quad . \tag{8.8}$$

Since, before DAM, the difficulty is not adjusted based on the adversary's strategy, the generation time of each block follows an exponential distribution

with rate λ. Therefore, at each step of mining a new block B, we sample a random number from an exponential distribution with rate λ to represent its generation time.

Security threshold (upcoming years). To analyze the security threshold of Bitcoin in upcoming years, we need an estimation of the time-reward equation of Bitcoin blocks. The reward-fee equation takes the form $\texttt{Reward}(t) = R + \texttt{fee}_0 + r_{\texttt{fee}}t$, where R is the protocol reward, t is the block generation time in minutes, $\texttt{fee}_0$ is the base fee, and $r_{\texttt{fee}}$ is the fee increase rate. We extracted the Bitcoin block transaction fee for all blocks mined during the period from January 2023 to December 2024. Then, for each day in this period, we used regression techniques to obtain the reward-fee equation based on different values of the protocol reward R. For each protocol reward R, we calculated the average value of the *block reward increase ratio* defined as $\frac{r_{\texttt{fee}}}{R+\texttt{fee}_0}$ among those promising days whose block reward increase ratio is in the top 10 percent. According to our analysis, the block reward increase ratio during promising days under the current protocol reward of 3.125 BTC, after 1, 2, 3, 4 protocol reward halvings, and under the eventual protocol reward of 0, can be estimated as 0.0091, 0.0150, 0.0230, 0.0336, 0.0459, and 0.2057, respectively. Figure 8.1 illustrates the security threshold for Bitcoin mining before difficulty adjustment based on the extracted block reward increase ratios, considering three different adversarial communication capabilities of 0.5, 0.75, and 0.9. Note that for the same block reward increase ratio, the security threshold for pre-DAM selfish mining is higher than that for the difficulty-adjusted case. For miners whose mining share exceeds the former threshold, selfish mining does not incur an initial loss period and becomes profitable immediately. However, for miners whose mining share falls between the two thresholds, selfish mining results in losses initially and becomes profitable only in the long term. We can conclude that in the coming years, as transaction fees gradually become the primary source of rewards, the threat of selfish mining becomes more significant. This is because not only does the minimum mining power required for a successful attack decrease, but the attacker would also gain immediate profits upon executing the attack.

Days susceptible to selfish mining. To identify the days most prone to selfish mining attacks, we compare the profits from selfish mining before difficulty adjustment on $\texttt{Day}_1$: October 8, 2023, with the time-fee equation $\texttt{fee}(t) = 0.0069t + 0.0359$, and on $\texttt{Day}_2$: December 17, 2023, with the time-fee equation $\texttt{fee}(t) = 0.0659t + 3.0457$, under two conditions: when the protocol reward is still substantial and after it has converged to zero. In Figure 8.7, we present the time-averaged profit a selfish miner can achieve before the difficulty adjustment on $\texttt{Day}_1$ and Day2, under both protocol rewards of 0 and 3.125 BTC, as a function of the selfish miner's mining share. This graph assumes a communication capability of 1 for the selfish miner and normalizes the time-averaged profit of the entire network under honest mining to 1. Selfish

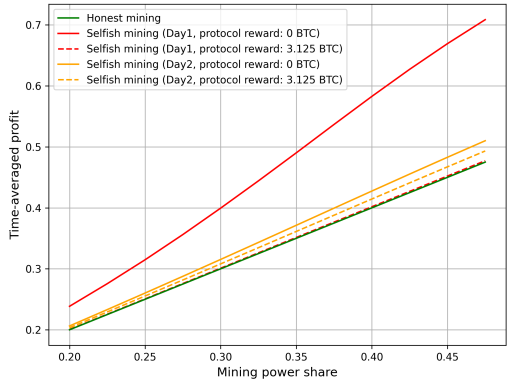


Figure 8.7: Time-averaged profit before difficulty adjustment.

mining becomes profitable on days when the block reward increase ratio $\frac{r_{\texttt{fee}}}{R+\texttt{fee}_0}$ is significant. To meet this condition with a substantial protocol reward, transaction fees must also be high and comparable to the protocol reward ($\texttt{Day}_2$). However, if the protocol reward becomes negligible, the Bitcoin network becomes more vulnerable to selfish mining on days when the rate of increase in transaction fees is high, even if those days do not coincide with high transaction fees per block ($\texttt{Day}_2$).

8.6 Enhanced Volatile Reward Model

Although the simplified volatile reward model introduced in Section 8.4 improves upon existing models [10, 12] by accounting for multiple fee levels, it still has inherent limitations that hinder precise analysis of Bitcoin mempool patterns. These limitations include:

- **Linear assumption on the time-fee relationship.** The fee accumulated over time by a block does not necessarily increase linearly. In practice, the time-fee relationship is often concave, suggesting that the rate of increase in block reward diminishes as generation time increases.
- **Block reward estimation based on a single dependent variable, "block generation time".** This assumption disregards the correlation between a block's transaction fee reward and the generation times of preceding blocks. In practice, the prolonged generation time of a block not only increases its own reward but also affects the rewards of subsequent blocks.

- **Imprecision in time-fee estimation due to sparsity of block data.** Using sparse block data for time-fee regression (on average, one data point every 10 minutes) can lead to reduced accuracy in modeling mempool behavior for a specified period, causing deviations between real-time patterns and the model's outcomes.

We elaborate further on the limitations of the simplified model in the Appendix 8.A5.

To address the limitations of the simplified model, this section introduces our Asynchronous Advantage Actor-Critic (A3C)-based implementation, designed to derive a near-optimal mining policy while accounting for the complexities of a real-world Bitcoin mempool. While implementing the Bitcoin environment, we observed that accurately modeling the mempool requires a significantly large state space. This large state space is needed to store fine-grained information about the mempool state, particularly with respect to any blocks involved in a fork race. This expanded state space makes traditional MDP tools impractical. Consequently, we adopt A3C—a reinforcement learning approach well-suited for high-dimensional environments. The A3C algorithm [20], introduced by DeepMind, is an actor–critic method [18] that leverages asynchronous parallelism. In A3C, multiple agents interact independently with separate instances of the environment and asynchronously update a shared global network. This parallelism not only accelerates exploration and learning but also enhances convergence stability in complex environments with high-dimensional state spaces, making A3C a superior option compared to traditional reinforcement learning methods.

A3C enables us to implement an *enhanced mempool*, which allows for a more accurate analysis of mining strategies in the transaction-fee era by overcoming the oversimplifications of earlier models. Specifically, the enhanced mempool incorporates polynomial and logarithmic regression techniques to better estimate the time-fee relationship. The mempool state is dynamically updated based on the passage of time and block generation events, capturing the impact of historical activity on the current state. Additionally, it leverages fine-grained mempool data to model the time-fee pattern more precisely, enabling shorter and more stable regression periods that mitigate sample deviation issues.

Our A3C-based implementation codes are provided in [7]. Additionally, a discussion on the implementation of the A3C-based tool is provided in Appendix 8.A6. This discussion includes a comprehensive description of the objective function (Appendix 8.A6.1) and the A3C implementation details (Appendices 8.A6.2 and 8.A6.3).

8.6.1 Enhanced Mempool Implementation

In the simplified model, we estimate the time-fee equation using only data on block transaction fees and generation times. To better estimate how transaction fees fluctuate over time, we can improve the model by incorporating the behavior of the Bitcoin mempool. The mempool is a storage space where unconfirmed transactions are stored before they are included in a block. To increase the chance of inclusion in a block, each Bitcoin transaction pays a fee to the block miner. The total fee that a transaction pays is calculated based on the transaction's weight, which is measured in virtual bytes (vBytes). Virtual bytes are a weight unit introduced by the SegWit upgrade. The fee is typically expressed in satoshis per virtual byte (sat/vByte), where each satoshi is 10^{-8} BTC. Each Bitcoin block has a size limit of 1 virtual megabyte (vMB) [3]. Miners prioritize filling the block with the most valuable transactions from the mempool, where the term *valuable* refers to transactions that offer the highest fees in satoshis per virtual byte. In the context of the enhanced volatile reward model, we define the *base fee* for a given mempool period as the fee for which there is at least 1 virtual megabyte of transactions offering that fee or higher at all times during that period.

Our implementation categorizes the transactions available in the mempool into N^{range} sat/vByte ranges, denoted as $\{\texttt{fee}_0, \texttt{fee}_1, \ldots, \texttt{fee}_{N^{\text{range}}-1}\}$, where the fee levels are ordered in ascending sequence. A transaction paying f sat/vByte, where $\texttt{fee}_j \leq f < \texttt{fee}_{j+1}$, is assigned to the range corresponding to $\texttt{fee}_j$. $\texttt{fee}_0$ denotes the base fee, implying that all transactions available in the mempool pay at least $\texttt{fee}_0$ sat/vByte. In our implementation, we assume an unlimited number of transactions paying the base fee. The weight of transactions in the range $\texttt{fee}_j$ increases over time according to a specific equation, called the *weight-time* equation, which is estimated based on the transaction weight growth in the range $\texttt{fee}_j$. Whenever a new block is mined, a total weight equivalent to 1 virtual megabyte is deducted from the mempool, starting from the highest sat/vByte ranges. To estimate the weight increase function for each sat/vByte range over a given time period, we analyzed mempool statistics during that period using the information presented in [8]. Then, we applied regression techniques to derive the weight-time equation for each sat/vByte range.

The estimated weight-time curves based on mempool statistics from December 17, 2023, are shown in Figure 8.8. To depict Figure 8.8, the transactions in mempool are categorized into five ranges of sat/vByte, with the base fee equal to 200 sat/vByte. This base fee is chosen because, at all times on December 17, 2023, there was at least 1 virtual MB of transactions in the mempool paying a fee higher than 200 sat/vByte. This indicates that, regardless of the block generation time, a block mined on this day could collect at least 200 million satoshis (2 BTC). An interesting observation from Figure 8.8 is that, although

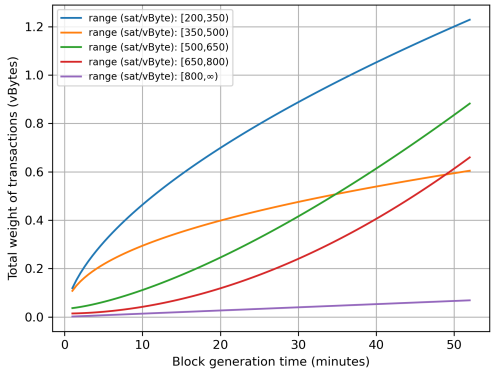


Figure 8.8: Total transaction weight for different ranges of sat/vByte as a function of block generation time (December 17, 2023).

the weights of transactions in all sat/vByte ranges are increasing over time, the rate of increase for weights in the lower sat/vByte ranges decreases over time (indicating a concave weight-time equation). In contrast, the rate of increase for weights in the higher sat/vByte ranges accelerates (indicating a convex weight-time equation). This observation aligns with the competition that exists in practice among transactions for inclusion in blocks. As the generation time for the next potential block increases, this competition intensifies, where newer transactions need to submit higher fees to maintain their chances of being included in the next block.

8.6.2 A3C Actions and State Representation

In our A3C-based implementation, we assume that adversary $\mathcal{A}$ mines on top of its secret chain, while the honest miners $\mathcal{H}$ mine on top of the single public chain. These two chains share a common subchain, after which they diverge from each other. The common subchain is referred to as the canonical chain, whose included blocks cannot later be orphaned by the adversarial chain. The adversary's chain (honest chain) proposed on top of the canonical chain tip is referred to as the adversarial fork (honest fork). Let $l_{\mathcal{A}}$ ($l_{\mathcal{H}}$) denote the adversarial fork length (the honest fork length). We first explain the set of possible actions that the adversary can take.

`override`: The action represents that the adversary publishes a sub-fork of its secret fork whose length is one block longer than the honest fork. As a result

of `override` action, the honest miners abandon their current fork and start mining on top of the longer, adversarial fork. This action is feasible if $l_{\mathcal{A}} > l_{\mathcal{H}}$.

`match`: The action represents that the adversary publishes a sub-fork of its secret fork whose length is equal to the honest fork. Action `match` results in a race between 2 same-height forks. To encourage altruistic miners to mine on top of the adversarial fork, the adversary should immediately publish its fork after an honest block is released. By doing so, the adversarial fork can be delivered earlier to some of the altruistic nodes, where the ratio of these nodes is determined based on the adversarial communication capability factor. However, petty-compliant miners do not necessarily choose a fork based solely on its delivery time. To incentivize these miners, the adversarial fork must leave transactions with higher total fees in the mempool compared to the honest fork. This means that the adversarial fork should have a lower total transaction fee than the honest fork to improve its chances of winning the same-height fork race in the presence of petty-compliant miners. Typically, if the adversary has mined the block of height l in its fork earlier than the block of height l in the honest fork, the adversarial fork up to and including length l contains a lower total transaction fee compared to the honest fork up to and including length l. This is because, during the time gap between mining the lth adversarial block and the lth honest block, several fee-rich transactions may arrive that honest miners can include in their blocks, which are absent from the adversarial fork up to and including block l. This action is feasible if $l_{\mathcal{A}} \geq l_{\mathcal{H}}$.

`wait`: This action represents that the adversary continues mining new blocks on top of its secret fork. This action is always feasible.

$\texttt{adopt}_i$: The action $\texttt{adopt}_i$, $i \in \{0, 1, \ldots, k_1\}$ represents that the adversary abandons its adversarial fork and starts mining a new fork on top the longest public chain whose latest i blocks are pruned. In the fixed block reward model, considering only the action $\texttt{adopt}_0$—namely adopting the longest chain—is enough, as $\texttt{adopt}_i$ for $i > 0$ cannot lead to a more profitable strategy. However, in the volatile block reward model, a deep block in the honest fork may have an extraordinary amount of transaction fees, incentivizing the adversary to orphan the block and steal its transaction fees. A block with an extraordinary fee can result from including a whale transaction, as considered in [10], or from having a long generation time. Action $\texttt{adopt}_i$ is feasible if $i \leq l_{\mathcal{H}}$.

$\texttt{undercut}_{\texttt{block}}$: This action represents that the adversary tries to undercut the tip block of the longest public chain denoted by $B_{\mathcal{H}}^{\text{tip}}$. To perform block undercutting, the adversary: i) abandons its fork, ii) adopts $l_{\mathcal{H}} - 1$ blocks in the honest fork, namely the entire honest fork except for the last block $B_{\mathcal{H}}^{\text{tip}}$, and iii) adjusts the total transaction fee in its potential next block to be less than the total transaction fee included in $B_{\mathcal{H}}^{\text{tip}}$. If, after taking the action $\texttt{undercut}_{\texttt{block}}$, the next block is mined by the adversary, the adversary's subsequent action is

`match`. Choosing action `match` will result in a race between two forks of the same height, differing only in their last block. In this scenario, the altruistic miners will continue mining on top of the honest fork since they received it earlier. However, the petty-compliant miners will switch to mining on top of the adversarial fork as it contains a lower amount of transaction fees. Our implementation considers the ϵ-petty-compliant miners with an adjustable ϵ, where the petty-compliant miners deviate from the honest strategy if doing so allows them to earn at least ϵ BTC more. If the amount of remaining transaction fee in the transaction pool is relatively low and the tip block $B_{\mathcal{H}}^{\text{tip}}$ includes a relatively large amount of transaction fee, undercutting block $B_{\mathcal{H}}^{\text{tip}}$ may be a logical action for the adversary as it enables the adversary to fill its block with transactions with higher fees. This action is always feasible.

$\texttt{undercut}_{\texttt{fork}}$: This action represents that the adversary tries to undercut the entire honest fork using its fork whose length is 1 less than the honest fork length. To perform fork undercutting, the adversary adjusts the transaction fee in its potential next block in a way that the total transaction fee in the adversarial fork becomes less than that in the honest fork. For $l_{\mathcal{H}} = 1$, this action is the same as $\texttt{undercut}_{\texttt{block}}$. Similar to the scenario explained for action $\texttt{undercut}_{\texttt{block}}$, if the next block is mined and published by the adversary, the petty-compliant miners are incentivized to mine on top of the adversarial fork. Under the presence of petty-compliant miners, action $\texttt{undercut}_{\texttt{fork}}$ increases the probability of orphaning the honest fork when the adversary's fork is lagging behind in the fork race. This action is feasible if $l_{\mathcal{A}} = l_{\mathcal{H}} - 1$.

A state at step t in our implementation represents a race between the honest and adversarial forks and has the following format:

$$
\begin{aligned}
s_t = \Big(& l_{\mathcal{A}}, l_{\mathcal{H}}, \texttt{Latest}, \texttt{Match}, \texttt{undercut}, \\
& BR_{\mathcal{A}} = \{BR_{\mathcal{A}}^1, BR_{\mathcal{A}}^2, \ldots, BR_{\mathcal{A}}^{\texttt{maxForkLen}}\}, \\
& \texttt{Pool}_{\mathcal{A}} = \{\texttt{Pool}_{\mathcal{A}}^1, \texttt{Pool}_{\mathcal{A}}^2, \ldots, \texttt{Pool}_{\mathcal{A}}^{\texttt{maxForkLen}}\}, \\
& \texttt{Pool}_{\mathcal{H}} = \{\texttt{Pool}_{\mathcal{H}}^1, \texttt{Pool}_{\mathcal{H}}^2, \ldots, \texttt{Pool}_{\mathcal{H}}^{\texttt{maxForkLen}}\}, \\
& \texttt{Pool}_{\mathcal{C}} \Big) \ .
\end{aligned}
\tag{8.9}
$$

In the state representation above, $l_{\mathcal{A}}$ ($l_{\mathcal{H}}$) denotes the adversarial (honest) fork length. `Latest` is a boolean variable that represents whether the latest mined block is honest or adversarial. If the last block is honest and $l_{\mathcal{A}} \geq l_{\mathcal{H}}$, the adversary can take action `match`. `Match` is a boolean variable that represents whether the same-height fork race resulting from the action `match` is resolved or not. If, after taking action `match`, the next block is honest, the fork race

is resolved. However, if the next block is adversarial, the fork race persists. `undercut` is a boolean variable that represents whether the next block mined after taking actions $\texttt{undercut}_{\texttt{block}}$ or $\texttt{undercut}_{\texttt{fork}}$ is adversarial or not. If the adversary manages to mine the next block, it should then take action `match` to publish the undercutting block or fork to the rational miners. $BR^i_{\mathcal{A}}$ for $i \in \{1, 2, \ldots, \texttt{maxForkLen}\}$ denotes the block reward of the ith block in the adversarial fork. This block reward includes both the protocol reward and the transaction fees collected from the included transactions. Note that `maxForkLen` is a variable that determines the maximum length of a fork in our implementation. $\texttt{Pool}^i_{\mathcal{A}}$ ($\texttt{Pool}^i_{\mathcal{H}}$) for $i \in \{1, 2, \ldots, \texttt{maxForkLen}\}$ denotes the mempool statistics w.r.t. the chain whose tip block is the $i^{\texttt{th}}$ adversarial (honest) block. Once a new block is mined, it includes some of the transactions available in the mempool, resulting in the removal of those transactions from the mempool and consequently a modification in the mempool statistics. A state must store the mempool statistics w.r.t. the $i^{\texttt{th}}$ adversarial (honest) block as miners may need to abandon their current forks and continue mining on top of the $i^{\texttt{th}}$ adversarial (honest) block as a result of a specific action. Mempool statistics is an N^{range}-element array that contains the weight of transactions available in each of the satoshi-per-byte ranges $\{\texttt{fee}_0, \texttt{fee}_1, \ldots, \texttt{fee}_{N^{\text{range}}-1}\}$. The weight of transactions corresponding to each satoshi-per-byte range $\texttt{fee}_j$ increases over time according to the specific weight-time equation defined for the range $\texttt{fee}_j$. As mentioned earlier, once a new block is mined, a total weight of 1 virtual megabyte is deducted from the transactions in the highest satoshi-per-byte ranges. However, if the adversary takes one of the actions $\texttt{undercut}_{\texttt{block}}$ or $\texttt{undercut}_{\texttt{fork}}$, the block may be filled with less than 1 virtual megabyte of transactions to leave some valuable transactions in the mempool for petty-compliant miners. $\texttt{Pool}^i_{\mathcal{A}}$ ($\texttt{Pool}^i_{\mathcal{H}}$) is an array containing N^{range} elements, where the jth element represents the weight of total transactions of the mempool that pay $\texttt{fee}_{j-1}$ satoshis per virtual byte. Finally, $\texttt{Pool}_{\mathcal{C}}$ denotes the mempool statistics w.r.t. the canonical chain.

8.6.3 Undercutting as a Continuous Action

The actions `override`, `adopt`, `match`, and `wait` can be considered discrete actions. This means that once a new block is mined in the system, the agent decides whether to take or not take these actions. However, the action `undercut` is more appropriately viewed as a continuous action. When dealing with continuous actions, the agent must decide on a value within a range that represents the intensity or duration of the action. In the case of `undercut`, the adversary must determine how long to continue undercutting the tip of the canonical chain. After a certain point, the adversary should stop undercutting and resume mining on top of the tip block. The continuous nature of the

`undercut` action arises because, depending on the available transaction fees in the mempool, undercutting may or may not be the best action to take. As discussed in Section 8.A1, the `undercut` action is profitable when the mempool w.r.t. the tip of the chain is relatively empty, leading the undercutter to mine on top of the parent block of the tip to steal the transaction fees from the tip block. Over time, as the transaction fees available in the mempool increase, a point is reached where the fees w.r.t. the tip of the chain are large enough that continuing the `undercut` action becomes unprofitable. At this point, the agent should abandon undercutting and instead mine on top of the tip block.

In an A3C implementation, the neural network outputs a single scalar value representing the value of the current state, along with a probability distribution over the possible actions. For discrete actions, this probability distribution is a vector where each element represents the probability of selecting a specific action given the current state. For continuous actions, the network outputs the parameters of a probability distribution over the continuous action [17]. In our implementation, the neural network outputs a vector representing the probability distribution of the actions introduced in Section 8.6.2. Additionally, it outputs two scalars representing the mean and variance of the duration of the undercutting action, which is modeled as a continuous action.

8.6.4 Results and Implications: Mining Attacks under an Enhanced Volatile Model

This section presents results from our A3C-based implementation [7], analyzing mining attacks under the mempool patterns observed during different periods. Information regarding our A3C-based tool implementation is provided in 8.A6.2. To compare the time-averaged profits of a mining strategy π, we use the concept of the percentage increase in the miner's time-averaged profit, defined as follows:

$$\texttt{PI}^{\mathcal{A}}(\pi) = \frac{\texttt{Profit}^{\mathcal{A}}(\pi) - \texttt{Profit}^{\mathcal{A}}(\pi^{\mathcal{H}})}{\texttt{Profit}^{\mathcal{A}}(\pi^{\mathcal{H}})} * 100\% \ , \tag{8.10}$$

where $\texttt{Profit}^{\mathcal{A}}(\pi^{\mathcal{H}})$ represents the time-averaged profit achieved by the adversary if all miners, including the adversary, follow the honest strategy.

In this section, we analyze the profitability of selfish mining before the difficulty adjustment, as well as the single-block undercutting attack. The results related to the profitability of selfish mining after the difficulty adjustment are presented in Appendix 8.A7.

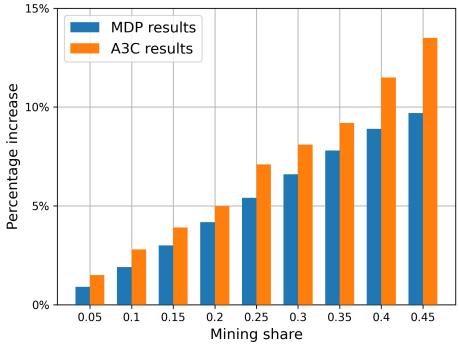


Figure 8.9: The percentage increase in time-averaged profit (December 18, 2023; protocol reward: 3.125 BTC).

8.6.4.1 Mining Attacks Before a Difficulty Adjustment

In Figure 8.9, the percentage increase in the adversary's time-averaged profit before a difficulty adjustment is shown as a function of the adversary's mining share. This figure is depicted based on the following configuration parameters: i) the protocol reward is set to 3.125 BTC, ii) the mempool pattern observed on December 18, 2023, between 23:00 and 24:00 CET, has occurred, iii) the adversary's communication capability equals 1. Figure 8.9 presents the results obtained from both our MDP implementation and A3C implementation. As shown in Figure 8.9, the strategy derived from the A3C implementation can outperform the strategy obtained from the MDP. Furthermore, as the adversary's mining share increases, the gap between the results obtained from these two approaches grows. As can be seen in Figure 8.9, even with a protocol reward of 3.125 BTC, on a day where the average transaction fee per block is comparable to the protocol reward (the average transaction fee per block on December 18, 2023, was 2.32 BTC [2]), the selfish mining attack can lead to profitability immediately after the attack begins.

Similarly, Figure 8.10 illustrates the percentage increase in the adversary's time-averaged profit before a difficulty adjustment, using the same assumptions as in Figure 8.9, with the exception that the protocol reward is set to zero. By comparing Figures 8.10 and 8.9, we observe that as the protocol reward decreases, deviations from the honest strategy become increasingly profitable, posing a potential threat to Bitcoin's progress in the near future.

Although Figure 8.10 shows that in the transaction-fee era, the same mempool pattern as occurred on December 18, 2023, can be concerning, there are some

other days where the threat of adversarial attack is more alarming. Figure 8.11 represents the percentage increase in profit before the difficulty adjustment from applying both selfish mining and undercutting attacks simultaneously under the mempool patterns of two different periods: December 18, 2023, between 23:00 and 24:00 CET (the same period as in Figures 8.10 and 8.9), and October 15, 2023, between 8:00 and 9:00 CET. The figure is depicted as a function of different ratios of petty-compliant miners under the following configuration parameters: i) the protocol reward equals zero, ii) the adversarial mining share is $\frac{1}{3}$, and iii) the adversary has a normal communication capability of 50%. As can be seen in Figure 8.11, even if all the honest miners are altruistic, the deviation still yields a substantial immediate profit for the adversary under the mempool pattern of October 15, 2023. By increasing the ratio of petty-compliant miners in the system, the deviation can become even more profitable. The main reason for the higher profitability of mining attacks under the mempool pattern of October 15, 2023, compared to that of December 18, 2023, is the low level of base fee available in the mempool. Between 8:00 and 9:00 CET on October 15, 2023, the base fee was 1 sat/vByte [8], implying that during this period, miners needed to fill at least some part of their blocks with transactions offering only 1 sat/vByte. In contrast, the base fee was 80 sat/vByte on December 18, 2023, between 23:00 and 24:00 CET [8]. This comparison shows that a higher base fee level for transactions available in the mempool can help mitigate mining attacks in the transaction-fee era.

In Appendix 8.A8, we present the selfish mining results obtained from our A3C implementation under the mempool behavior observed on two additional days.

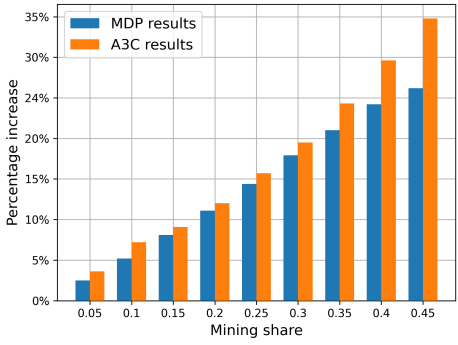


Figure 8.10: The percentage increase in time-averaged profit (December 18, 2023; protocol reward: 0 BTC).

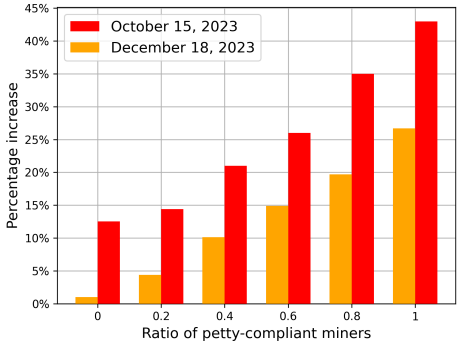


Figure 8.11: The percentage increase in time-averaged profit (protocol reward: 0 BTC).

8.6.4.2 Single Block Undercutting Attack

As discussed in Section 8.6.3, in our A3C-based implementation, we treat the undercut action as a continuous action, where the agent must not only decide whether to take this action but also determine the duration for which it should perform undercutting. Once the designated time has passed, the agent should stop undercutting the tip block and begin mining on top of it, as a sufficient number of transactions will have arrived in the system. To measure the profitability of the single-block undercutting attack, we have limited the adversary's possible action set, as outlined in Appendix 8.A6.3. Figure 8.12 shows the percentage increase in time-averaged profit achieved by performing the undercutting attack. The figure is depicted based on mempool patterns observed over different days between 8:00 and 9:00 CET, under the following configuration parameters: i) the protocol reward is set to 0, ii) the adversarial mining share and communication capability are set to $\frac{1}{3}$ and 50%, respectively, and iii) all non-adversarial miners are assumed to be petty-compliant.

Lower base fees make the Bitcoin blockchain more vulnerable to undercutting. This is especially true when the tip block has a long generation time, as it stays exposed to attack longer while the mempool builds up enough fees.

8.7 Conclusion

In this paper, we analyzed the impact of volatility in block rewards on mining strategies in Bitcoin. Under the fixed block reward model, which is primarily suitable for modeling Bitcoin's behavior during its first decade of existence, mining attacks such as selfish mining need to continue for at least one difficulty

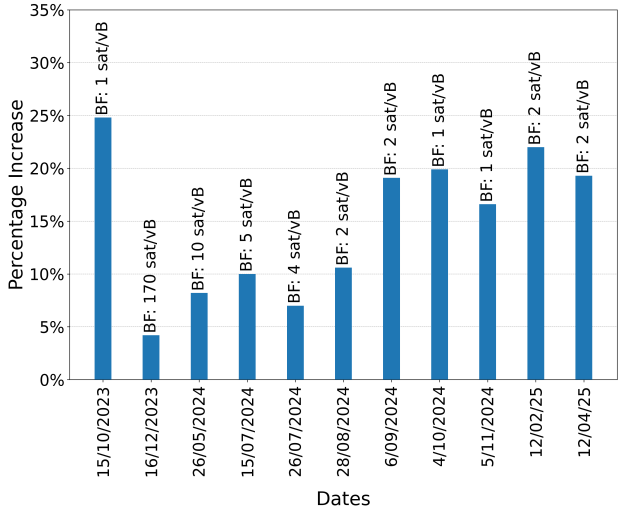


Figure 8.12: The profit percentage increase of undercutting. BF denotes the base fee.

epoch (approximately two weeks) to become profitable. However, using historical Bitcoin statistics, we have demonstrated that under a volatile block reward model, when transaction fees become a major source of incentive, adversarial strategies can be profitable immediately, bypassing the initial loss period. The threat of deviant strategy profitability is concerning as the protocol reward diminishes to zero, and even under the current protocol reward, on days when average fees per block are comparable to the protocol reward.

Once assigning a transaction fee for their transactions, Bitcoin users should view their paid fee not only as a reward to compensate miners but also as a factor that can significantly impact the security of their valued blockchain. To ensure the secure progress of Bitcoin, users should provide a stable source of incentives for miners, so they have no, or at least minimal, motivation to deviate from the Bitcoin protocol.

Bibliography

[1] Werlman codes. https://github.com/roibarzur/pto-selfish-mining, 2022.

[2] Bitcoin explore. https://btc.com/btc/blocks, 2024.

[3] Bitcoin wiki. https://en.bitcoin.it/wiki/Bitcoin, 2024.

[4] Coin market cap. https://coinmarketcap.com/, 2024.

[5] Gdp by country. https://www.worldometers.info/gdp/gdp-by-country/, 2024.

[6] Historical data on block transaction fees and generation times. https://blockchair.com/, 2024.

[7] Implementation of our mdp-based and a3c-based tools to analyze bitcoin mining under volatile block rewards. https://github.com/bitcoinVolatileReward/bitcoinVolatileReward, 2024.

[8] Johoe's bitcoin mempool statistics. https://jochen-hoenicke.de/queue/, 2024.

[9] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8:1–74, 2021.

[10] Roi Bar-Zur, Ameer Abu-Hanna, Ittay Eyal, and Aviv Tamar. Werlman: To tackle whale (transactions), go deep (rl). In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 93–110. IEEE, 2023.

[11] Jonah Brown-Cohen, Arvind Narayanan, Alexandros Psomas, and S Matthew Weinberg. Formal barriers to longest-chain proof-of-stake protocols. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 459–473, 2019.

[12] Miles Carlsten, Harry Kalodner, S Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 154–167, 2016.

[13] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102, 2018.

[14] Tiantian Gong, Mohsen Minaei, Wenhai Sun, and Aniket Kate. Towards overcoming the undercutting problem. In *International Conference on Financial Cryptography and Data Security*, pages 444–463. Springer, 2022.

[15] Cyril Grunspan and Ricardo Pérez-Marco. On profitability of selfish mining. *arXiv preprint arXiv:1805.08281*, 2018.

[16] Cyril Grunspan and Ricardo Pérez-Marco. Profit lag and alternate network mining. In *The International Conference on Mathematical Research for Blockchain Economy*, pages 115–132. Springer, 2023.

[17] Yanwei Jia and Xun Yu Zhou. Policy gradient and actor-critic learning in continuous time and space: Theory and algorithms. *Journal of Machine Learning Research*, 23(275):1–50, 2022.

[18] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[19] Ilya Kostrikov. Pytorch implementations of asynchronous advantage actor critic. https://github.com/ikostrikov/pytorch-a3c, 2018.

[20] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[21] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Satoshi Nakamoto*, 2008.

[22] Kevin Alarcón Negy, Peter R Rizun, and Emin Gün Sirer. Selfish mining re-examined. In *International Conference on Financial Cryptography and Data Security*, pages 61–78. Springer, 2020.

[23] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *Financial Cryptography and Data Security: 20th International Conference, FC 2016, Christ Church, Barbados, February 22–26, 2016, Revised Selected Papers 20*, pages 515–532. Springer, 2017.

[24] Roozbeh Sarenche, Svetla Nikova, and Bart Preneel. Deep selfish proposing in longest-chain proof-of-stake protocols. *Cryptology ePrint Archive*, 2024.

[25] Roozbeh Sarenche, Ren Zhang, Svetla Nikova, and Bart Preneel. Selfish mining time-averaged analysis in bitcoin: Is orphan reporting an effective countermeasure? *IEEE Transactions on Information Forensics and Security*, 2024.

[26] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[27] Itay Tsabary and Ittay Eyal. The gap game. In *Proceedings of the 2018 ACM SIGSAC conference on Computer and Communications Security*, pages 713–728, 2018.

[28] Roi Bar Zur, Ittay Eyal, and Aviv Tamar. Efficient mdp analysis for selfish-mining in blockchains. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 113–131, 2020.

Appendix

8.A1 Mining Attacks

8.A1.1 Selfish Mining

In the selfish mining attack, introduced by Eyal et al. in [13], a selfish miner withholds a newly mined block B_1 instead of immediately publishing it to the network. By keeping B_1 secret, the selfish miner creates a private fork one block ahead of the public chain. While other miners continue mining on the public chain, the selfish miners work to extend their private forks. If a selfish miner mines a second block B_2 on top of its private fork, its chain gains a two-block lead over the public chain. When an honest miner mines a new block B_3 on top of the public chain, the selfish miner can reveal its secret fork. Since the selfish miner's chain, extended by blocks B_1 and B_2, is longer compared to the public chain, extended by block B_3, the selfish miner's chain becomes the canonical chain. Consequently, block B_3, mined by honest miners, becomes orphaned and is excluded from the canonical chain. However, it is not necessarily the case that the selfish miner can always achieve a two-block lead. If the selfish miner keeps its block secret and honest miners mine the next block, a fork race occurs between two forks of the same height, with only one fork becoming the canonical chain. The probability of winning this race depends on how quickly the selfish miner can propagate its blocks through the network, encouraging other miners to build on top of the selfish miner's fork. This attack can increase the selfish miner's block reward relative to other miners. In [23], the authors implemented an MDP-based tool to determine the optimal selfish mining strategy that a selfish miner can follow in a setting where the reward for all blocks is the same, referred to as the fixed block reward model. In [15], the authors argued that the block ratio is not a good benchmark for analyzing selfish mining profitability and that one should use the time-averaged profit definition for this analysis. They demonstrated that selfish mining during the first difficulty epoch cannot be more profitable than honest mining, resulting in an initial loss period for selfish mining. However, from the subsequent epoch onward, the attack becomes profitable as the difficulty decreases due to the selfish mining attack. To reduce the initial loss period, the authors in [22] introduced intermittent selfish mining, where the selfish miner alternates between one epoch of selfish mining and one epoch of honest mining. The authors in [16] introduced the term *profit lag* to

describe the point at which the attack consistently becomes more profitable than honest mining in order to analyze the initial period of loss. They argued that although intermittent selfish mining's profitability may exceed that of honest mining at certain points in the early cycles of the attack, its profit lag is actually longer than the profit lag of standard selfish mining. In [25], the authors introduced smart intermittent selfish mining, where the selfish miner performs selfish mining for half of each block. Smart intermittent selfish mining can outperform the intermittent selfish mining introduced in [22] and reduces the profit lag. The selfish mining attack is not specific to the Bitcoin network and can be applied to any longest-chain-based blockchain. As analyzed in [11, 24], selfish mining can be even more destructive in the context of longest-chain Proof-of-Stake protocols due to factors such as proposer predictability and the nothing-at-stake phenomenon.

8.A1.2 Undercutting

The undercutting attack, introduced by Carlsten et al. in [12], can be profitable in scenarios where a significant portion of the mining power is petty-compliant. Petty-compliant miners are those who may deviate from the honest mining strategy to earn higher profits. As the protocol reward approaches zero, the undercutting attack poses an increasingly serious threat to the stability of the Bitcoin network. In this attack, when an undercutter becomes aware of a new block B_1 that extends the longest chain, it may choose not to accept it. If B_1 contains the majority of the transaction fees available in the mempool, the undercutter might be disincentivized from mining a relatively empty block on top of it. Instead, the undercutter may opt to mine on the parent of B_1 to create a competing block. The key aspect of the undercutting attack is that the undercutter should include fewer transaction fees in the competing block compared to B_1, leaving a substantial portion of the fees in the mempool for the future block. This strategy of generously leaving some transactions is crucial for the attack's success because if the undercutter successfully mines the competing block B_2, a fork will occur in the network between blocks B_1 and B_2. In the event of such a fork, petty-compliant miners would prefer to mine on top of the block that offers the most available transaction fees, rather than the oldest block——in this case, the competing block B_2. Attracting the mining power of rational miners to B_2 increases the undercutter's chances of winning the fork. If the next block is mined on top of the competing block B_2, the undercutter has successfully undercut block B_1. Carlsten et al. find that undercutting can become the equilibrium strategy for miners, leading to instability as miners undercut each other. However, this result is based on a model disregarding the block size limit. In paper [14], the authors have analyzed the undercutting attack with consideration of the block size limit and derived closed-form conditions on

the pending transaction set in the mempool that make undercutting profitable. They have also proposed an alternative transaction selection rule to counter undercutting; instead of fitting all available transactions into a block, miners include only a portion of the pending transaction fees to mitigate the risk of being undercut.

8.A2 Non-Predictable WeRLman

In order to modify the original WeRLman environment to a non-predictable version, we updated the function `tryGetWithAndWithoutNewFee` in the file `bitcoin_simplified_fee_model.py`, which is available in the repository referenced in [1], as presented in Algorithm 1.

8.A3 Adversarial Strategies in the Original and Non-predictable WeRLman Environments

Let α, g, p, and F denote the adversarial mining share, the adversarial communication capability, the whale transaction probability, and the extra fee of a whale transaction, respectively. We denote by $N_{\mathcal{A}}$, $N_{\mathcal{H}}$, and $R_{\mathcal{A}}$ the normalized average number of canonical adversarial blocks, the normalized average number of canonical honest blocks, and the normalized average adversarial block reward. The time-averaged profit of strategy π can be calculated as $\texttt{Profit}(\pi) = \frac{R_{\mathcal{A}}(\pi)}{N_{\mathcal{A}}(\pi)+N_{\mathcal{H}}(\pi)}$. Note that the time-averaged profit of the adversary if mining honestly is equal to $\texttt{Profit}(\pi^{\mathcal{H}}) = \alpha(1 + Fp)$.

We introduce three strategies: π_1^{WeRLman}, $\pi_1^{\text{Non-predictable}}$, and $\pi_2^{\text{Non-predictable}}$. Each strategy specifies the adversary's action in each state of the fork race. The adversary takes these actions upon the event of block mining. A newly mined block is adversarial with probability α and honest with probability $1 - \alpha$. At each event of mining a block at a *new* height, a transaction is sampled: it is a whale transaction with probability p and a normal transaction with probability $1 - p$. Note that in the original WeRLman environment (used by strategy π_1^{WeRLman}), the sampled transaction is added to the mempool and included in the next block. In contrast, in the non-predictable WeRLman environment (used by strategies $\pi_1^{\text{Non-predictable}}$ and $\pi_2^{\text{Non-predictable}}$), the sampled transaction is assumed to have already been included in the newly mined block. In a same-height fork race between a published adversarial fork and an honest fork, in addition to the adversarial mining power, a fraction g of honest miners will also mine on top of the adversarial fork.

Algorithm 1 Updated function `tryGetWithAndWithoutNewFee`

```
Input: Current state CS (self), and previous state PS
Output: Updated states new_with_fee, new_without_fee
current_max_fork ← max(CS.h, CS.a)
prev_max_fork ← max(PS.h, PS.a)
new_with_fee ← copy of CS
new_without_fee ← copy of CS
if prev_max_fork < current_max_fork and CS.pool < max_pool then
    new_with_fee.pool ← new_with_fee.pool +1
    if (CS.h − PS.h) > 0 then
        new_honest ← True
    else
        new_honest ← False
    end if
    if new_honest then
        if CS.T_h = PS.T_h then
            with_fee ← 1
        else
            with_fee ← 0
        end if
        new_with_fee.T_h ← new_with_fee.T_h + with_fee
    else
        if a ≤ h then
            if CS.T_a = PS.T_a then
                with_fee ← 1
            else
                with_fee ← 0
            end if
            new_with_fee.T_a ← new_with_fee.T_a + with_fee
        else
            new_with_fee.L[-1] ← 1
        end if
    end if
end if
return new_with_fee, new_without_fee
```

We use four actions to describe the strategies: `adopt`, denoting acceptance of the honest fork as the canonical chain; `wait`, denoting continued mining on top of the adversarial fork; `publish`, denoting the release of a longer adversarial fork; and `match`, denoting the release of an adversarial fork at the same height as the honest fork immediately after the publication of a new honest block.

8.A3.1 Strategy π_1^{WeRLman}

We use a tuple $(n_{\mathcal{A}}, n_{\mathcal{H}}, \texttt{whale}^{\texttt{pool}}, \texttt{whale}^{\texttt{block}})$ to represent a state during a fork race. Here, $n_{\mathcal{A}}$ and $n_{\mathcal{H}}$ denote the number of blocks in the adversarial and honest forks, respectively. The variable $\texttt{whale}^{\texttt{pool}}$ indicates the type of transaction currently in the pool w.r.t. the adversarial fork, which can be included in the next block: $\texttt{whale}^{\texttt{pool}} = 1$ if there is a whale transaction, and $\texttt{whale}^{\texttt{pool}} = 0$ if there is a normal transaction. $\texttt{whale}^{\texttt{block}}$ indicates the type of transaction included in the adversarial block: $\texttt{whale}^{\texttt{block}} = 1$ if the block includes a whale transaction, and $\texttt{whale}^{\texttt{block}} = 0$ if it includes a normal transaction.

We define four states as follows:

- State $S_{0,0} : (n_{\mathcal{A}} = 0, n_{\mathcal{H}} = 0, \texttt{whale}^{\texttt{pool}} = 0, \texttt{whale}^{\texttt{block}} = \perp)$. This state denotes that both the adversary and honest miners are mining on top of the same canonical chain, and the existing transaction in the mempool is normal.
- State $S'_{0,0} : (n_{\mathcal{A}} = 0, n_{\mathcal{H}} = 0, \texttt{whale}^{\texttt{pool}} = 1, \texttt{whale}^{\texttt{block}} = \perp)$. This state denotes that both the adversary and honest miners are mining on top of the same canonical chain, but the existing transaction in the mempool is a whale transaction.
- State $S_{1,0} : (n_{\mathcal{A}} = 1, n_{\mathcal{H}} = 0, \texttt{whale}^{\texttt{pool}} = 1, \texttt{whale}^{\texttt{block}} = 0)$. This state denotes that the secret adversarial fork contains a single block with a normal transaction, while a whale transaction remains in the mempool to be included in the next block.
- State $S_{1,1} : (n_{\mathcal{A}} = 1, n_{\mathcal{H}} = 1, \texttt{whale}^{\texttt{pool}} = 1, \texttt{whale}^{\texttt{block}} = 0)$. This state denotes that there is a fork race between the adversarial and honest forks, both with a height of one block. The adversarial block includes a normal transaction, while the honest block includes a whale transaction. The next block in the adversarial fork can steal the whale transaction from the honest block. Note that upon transferring to this state, a new transaction is not sampled, as the same height block already exists.

Strategy π_1^{WeRLman} is introduced in Table 8.1.

Let $P_{0,0}$, $P'_{0,0}$, $P_{1,0}$, $P_{1,1}$ denote the stationary probability of being in states $S_{0,0}$, $S'_{0,0}$, $S_{1,0}$, $S_{1,1}$, respectively. We have:

$$\begin{aligned} P_{0,0} &= \frac{1-p}{1+p(1-p)\alpha(1-\alpha)}, \quad P_{1,0} = \frac{\alpha p(1-p)}{1+p(1-p)\alpha(1-\alpha)}, \\ P_{1,1} &= \frac{p(1-p)\alpha(1-\alpha)}{1+p(1-p)\alpha(1-\alpha)}, \quad P'_{0,0} = 1 - P_{0,0} - P_{1,0} - P_{1,1}. \end{aligned} \tag{8.11}$$

Table 8.1: Strategy π_1^{WeRLman}.

Initial State	Action	Probability	Final State	Reward
$S_{0,0}$	`wait`. If the next block is honest, then switch to `adopt`. If the next block is adversarial and the next transaction is normal, then switch to `publish`.	αp	$S_{1,0}$	0
		$\alpha(1-p)$	$S_{0,0}$	1
		$(1-\alpha)p$	$S'_{0,0}$	0
		$(1-\alpha)(1-p)$	$S_{0,0}$	0
$S'_{0,0}$	`wait`. If next block is honest, then switch to `adopt`. If the next block is adversarial, then switch to `publish`.	αp	$S'_{0,0}$	$1+F$
		$\alpha(1-p)$	$S_{0,0}$	$1+F$
		$(1-\alpha)p$	$S'_{0,0}$	0
		$(1-\alpha)(1-p)$	$S_{0,0}$	0
$S_{1,0}$	`wait`. If the next block is adversarial, `wait` until $n_{\mathcal{A}} - n_{\mathcal{H}} = 1$, then switch to `publish`.	αp	$S'_{0,0}$	$2+F+(1+pF)\frac{\alpha}{1-2\alpha}$
		$\alpha(1-p)$	$S'_{0,0}$	$2+F+(1+pF)\frac{\alpha}{1-2\alpha}$
		$(1-\alpha)$	$S_{1,1}$	0
$S_{1,1}$	`match`	αp	$S'_{0,0}$	$2+F$
		$\alpha(1-p)$	$S_{0,0}$	$2+F$
		$(1-\alpha)gp$	$S'_{0,0}$	1
		$(1-\alpha)g(1-p)$	$S_{0,0}$	1
		$(1-\alpha)(1-g)p$	$S'_{0,0}$	0
		$(1-\alpha)(1-g)(1-p)$	$S_{0,0}$	0

The normalized numbers of canonical honest blocks, canonical adversarial blocks, and the normalized adversarial block reward achieved under strategy π_1^{WeRLman} can be obtained as follows:

$$\begin{aligned}
N_{\mathcal{H}} =& P_{0,0}(1-\alpha) + 2P_{1,1}(1-\alpha)(1-g) \\
&+ P_{1,1}(1-\alpha)g + P'_{0,0}(1-\alpha), \\
N_{\mathcal{A}} =& P_{0,0}\alpha(1-p) + \left(2 + \frac{\alpha}{1-2\alpha}\right) P_{1,0}\alpha + 2P_{1,1}\alpha \\
&+ P_{1,1}(1-\alpha)g + P'_{0,0}\alpha, \\
R_{\mathcal{A}} =& P_{0,0}\alpha(1-p) + P'_{0,0}\alpha(1+F) + P_{1,1}\alpha(2+F) + P_{1,1}(1-\alpha)g \\
&+ P_{1,0}\alpha(2+F) + P_{1,0}\alpha\frac{\alpha}{1-2\alpha}(1+pF) \quad .
\end{aligned} \tag{8.12}$$

8.A3.2 Strategy $\pi_1^{\text{Non-predictable}}$

We represent a state during a fork race using the following tuple: $(n_{\mathcal{A}}, n_{\mathcal{H}}, \texttt{whale}^{\texttt{block}})$. The descriptions of these parameters are the same as those provided in Appendix 8.A3.1. We define three states as follows:

- State $S_{0,0} : (n_{\mathcal{A}} = 0, n_{\mathcal{H}} = 0, \texttt{whale}^{\texttt{block}} = \perp)$. This state denotes that both the adversary and honest miners are mining on top of the same canonical chain.

Table 8.2: Strategy $\pi_1^{\text{Non-predictable}}$.

Initial State	Action	Probability	Final State	Reward
$S_{0,0}$	`wait`. If the next block is honest, then switch to `adopt`. If the next block is adversarial with a whale, then `publish`.	αp	$S_{0,0}$	$1+F$
		$\alpha(1-p)$	$S_{1,0}$	0
		$(1-\alpha)$	$S_{0,0}$	0
$S_{1,0}$	`wait`. If the next block is adversarial, `wait` until $n_{\mathcal{A}} - n_{\mathcal{H}} = 1$, then switch to `publish`.	α	$S_{0,0}$	$1+(1+pF)\left(1+\frac{\alpha}{1-2\alpha}\right)$
		$(1-\alpha)$	$S_{1,1}$	0
$S_{1,1}$	`match`	αp	$S_{0,0}$	$2+F$
		$\alpha(1-p)$	$S_{0,0}$	2
		$(1-\alpha)g$	$S_{0,0}$	1
		$(1-\alpha)(1-g)$	$S_{0,0}$	0

- State $S_{1,0} : (n_{\mathcal{A}} = 1, n_{\mathcal{H}} = 0, \texttt{whale}^{\texttt{block}} = 0)$. This state denotes that the secret adversarial fork contains a single block with a normal transaction.
- State $S_{1,1} : (n_{\mathcal{A}} = 1, n_{\mathcal{H}} = 1, \texttt{whale}^{\texttt{block}} = 0)$. This state denotes that there is a fork race between the adversarial and honest forks, both with a height of one block. The adversarial block includes a normal transaction. Note that upon transferring to this state, a new transaction is not sampled, as the same height block already exists.

Strategy $\pi_1^{\text{Non-predictable}}$ is presented in Table 8.2.

Let $P_{0,0}$, $P_{1,0}$, $P_{1,1}$ denote the stationary probability of being in states $S_{0,0}$, $S_{1,0}$, and $S_{1,1}$, respectively. We have:

$$\begin{aligned} P_{0,0} &= \frac{1}{1+(1-p)\,\alpha\,(2-\alpha)}, \quad P_{1,0} = \alpha\,(1-p)\,P_{0,0}, \\ P_{1,1} &= (1-\alpha)\,P_{1,0}. \end{aligned} \tag{8.13}$$

The normalized numbers of canonical honest blocks, canonical adversarial blocks, and the normalized adversarial block reward achieved under strategy $\pi_1^{\text{Non-predictable}}$ can be obtained as follows:

$$\begin{aligned} N_{\mathcal{H}} =& P_{0,0}\,(1-\alpha) + P_{1,1}\,(1-\alpha)\,g + 2P_{1,1}\,(1-\alpha)\,(1-g)\,, \\ N_{\mathcal{A}} =& P_{0,0}\alpha p + P_{1,0}\alpha\left(2+\frac{\alpha}{1-2\alpha}\right) + 2P_{1,1}\alpha + P_{1,1}\,(1-\alpha)\,g, \\ R_{\mathcal{A}} =& P_{0,0}\alpha p\,(1+F) + P_{1,0}\alpha\left(1+(1+pF)\left(1+\frac{\alpha}{1-2\alpha}\right)\right) \\ &+ P_{1,1}\,(1-\alpha)\,g + P_{1,1}\alpha\,(2+pF)\quad . \end{aligned} \tag{8.14}$$

8.A3.3 Strategy $\pi_2^{\text{Non-predictable}}$

We represent a state during a fork race using the following tuple: $(n_{\mathcal{A}}, n_{\mathcal{H}}, \texttt{whale}_{\mathcal{H}}^{\texttt{block}})$. Here, $n_{\mathcal{A}}$ and $n_{\mathcal{H}}$ denote the number of blocks in the adversarial and honest forks, respectively. $\texttt{whale}_{\mathcal{H}}^{\texttt{block}}$ indicates the type of transaction included in the first block of the honest fork: $\texttt{whale}_{\mathcal{H}}^{\texttt{block}} = 1$ if the block includes a whale transaction, and $\texttt{whale}_{\mathcal{H}}^{\texttt{block}} = 0$ if it includes a normal transaction. We define the states as follows:

- State $S_{0,0} : (n_{\mathcal{A}} = 0, n_{\mathcal{H}} = 0, \texttt{whale}_{\mathcal{H}}^{\texttt{block}} = \perp)$. This state denotes that both the adversary and honest miners are mining on top of the same canonical chain.
- State $S_{0,1} : (n_{\mathcal{A}} = 0, n_{\mathcal{H}} = 1, \texttt{whale}_{\mathcal{H}}^{\texttt{block}} = 1)$. This state denotes that the honest fork contains a single block with a whale transaction.
- State $S_{i,i} : (n_{\mathcal{A}} = i, n_{\mathcal{H}} = i, \texttt{whale}^{\texttt{block}} = 1)$ for $i \geq 1$. This state denotes that there is a fork race between the adversarial and honest forks, both with a height of i blocks. The first block in the honest fork includes a whale. The action `match` is not feasible in this state, as the ith adversarial block is mined after the ith honest block. Note that upon transferring to this state, a new transaction is not sampled, as the same height block already exists.
- State $S_{i,i+1} : (n_{\mathcal{A}} = i, n_{\mathcal{H}} = i + 1, \texttt{whale}^{\texttt{block}} = 1)$ for $i \geq 1$. This state denotes that there is a fork race between the adversarial fork of height i blocks with an honest fork of height $i + 1$ blocks. The first block in the honest fork includes a whale.

Strategy $\pi_2^{\text{Non-predictable}}$ is presented in Table 8.3. Let $P_{0,0}$, $P_{0,1}$, $P_{i,i}$, $P_{i,i+1}$ denote the stationary probability of being in states $S_{0,0}$, $S_{0,1}$, $S_{i,i}$, $S_{i,i+1}$, respectively. We have:

$$\begin{aligned} P_{0,0} &= \frac{1-(1-\alpha)(\alpha+p)}{1-(1-p)\alpha(1-\alpha)}, \quad P_{0,1} = \frac{(1-\alpha)p(1-\alpha(1-\alpha))}{1-(1-p)\alpha(1-\alpha)}, \\ \sum_{i=1}^{\infty} P_{i,i} &= P_{0,1}\frac{\alpha}{1-\alpha(1-\alpha)}, \quad \sum_{i=1}^{\infty} P_{i,i+1} = P_{0,1}\frac{\alpha(1-\alpha)}{1-\alpha(1-\alpha)}. \end{aligned} \tag{8.15}$$

The normalized numbers of canonical honest blocks, canonical adversarial blocks, and the normalized adversarial block reward achieved under strategy $\pi_2^{\text{Non-predictable}}$ can be obtained as follows:

Table 8.3: Strategy $\pi_2^{\text{Non-predictable}}$.

Initial State	Action	Probability	Final State	Reward
$S_{0,0}$	`wait`. If the next block is adversarial, then switch to `publish`. If the next block is honest with a normal transaction, then `adopt`.	αp	$S_{0,0}$	$1+F$
		$\alpha(1-p)$	$S_{0,0}$	1
		$(1-\alpha)p$	$S_{0,1}$	0
		$(1-\alpha)(1-p)$	$S_{0,0}$	0
$S_{0,1}$	`wait`. If the next block is honest with a normal transaction, then `adopt` the whole honest fork. If the next block is honest with a whale transaction, then `adopt` except the last honest block.	α	$S_{1,1}$	0
		$(1-\alpha)p$	$S_{0,1}$	0
		$(1-\alpha)(1-p)$	$S_{0,0}$	0
$S_{i,i}$ $(i \geq 1)$	`wait`. If the next block is adversarial, then `publish`.	α	$S_{0,0}$	$1+F+i(1+pF)$
		$(1-\alpha)$	$S_{i,i+1}$	0
$S_{i,i+1}$ $(i \geq 1)$	`wait`. If the next block is honest with a normal transaction, then `adopt` the whole honest fork. If the next block is honest with a whale transaction, then `adopt` except the last honest block.	α	$S_{i+1,i+1}$	0
		$(1-\alpha)p$	$S_{0,1}$	0
		$(1-\alpha)(1-p)$	$S_{0,0}$	0

$$
\begin{aligned}
N_{\mathcal{H}} &= P_{0,0}(1-\alpha)(1-p) + P_{0,1}(1-\alpha)p + 2P_{0,1}(1-\alpha)(1-p) \\
&\quad + \frac{P_{0,1}\alpha(1-\alpha)^2}{(1-\alpha(1-\alpha))^2} + \frac{P_{0,1}(2-p)\alpha(1-\alpha)^2}{1-\alpha(1-\alpha)} \ , \\
N_{\mathcal{A}} &= \left(\frac{P_{0,1}\alpha^2}{1-\alpha(1-\alpha)}\right)\left(\frac{1}{1-\alpha(1-\alpha)}+1\right) + P_{0,0}\alpha \ , \\
R_{\mathcal{A}} &= \left(\frac{P_{0,1}\alpha^2}{1-\alpha(1-\alpha)}\right)\left(\frac{(1+pF)}{1-\alpha(1-\alpha)}+(1+F)\right) + P_{0,0}\alpha(1+pF) \ .
\end{aligned}
\tag{8.16}
$$

8.A4 Simplified Mempool Environment: Supplementary

Deriving a time-fee equation compatible with the Bitcoin environment is challenging, as transaction fees vary across different periods due to various reasons such as fluctuations in Bitcoin's supply and demand dynamics. This variability indicates that the optimal mining strategy for an adversary can differ significantly across different periods. To maximize its profit, the adversary must accurately predict the time-fee equation for the targeted attack period.

For the simplified mempool implementation, we extracted historical data [6] on total transaction fees and block generation times for Bitcoin blocks mined across different periods. Using regression techniques, we derived the time-fee equation for the target period. This time-fee equation enables us to estimate

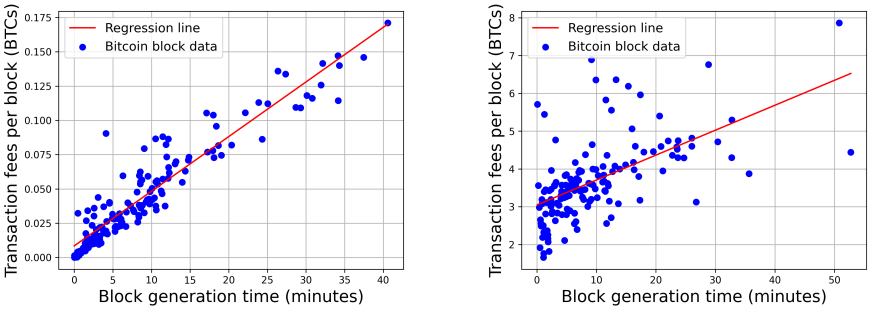
the reward function and analyze the profitability of mining strategies. As an example, in Figure 8.13, the total transaction fee of a block in BTC, denoted by `fee`, is depicted as a function of its generation time in minutes, denoted by t, for two different periods. Each point represents a block, and the line represents the time-fee equation. Figure 8.13a corresponds to Bitcoin blocks mined on January 1, 2023, with relatively low transaction fees per block, while Figure 8.13b corresponds to blocks mined on December 17, 2023, during a period of high transaction fees.

As already mentioned in 8.4.2, a linear time-fee equation can be expressed as $\texttt{fee}(t) = \texttt{fee}_0 + r_{\texttt{fee}}t$, where $\texttt{fee}_0$ and $r_{\texttt{fee}}$ are base fee and fee increase rate, respectively. At $t = 0$, the transaction reward of the next potential block is equal to the base fee. This reflects reality, as there are always some low-value transactions in the mempool that miners can use to fill their blocks. As block generation time increases, the collected transaction fee rises by the fee increase rate for each additional unit of time due to the arrival of new transactions.

We assume that the time-fee equation is a linear function, and time is divided into $M \geq 2$ discrete steps of the form $\{t_i = i\Delta \mid i \in [M]\}$, Δ denotes the length of each time step. For a block with block generation time t, the transaction fee reward can be obtained as follows:

$$\texttt{fee}(t) = \begin{cases} \texttt{fee}(i\Delta), & \text{if} \quad i\Delta \leq t < (i+1)\Delta \quad \text{for} \quad i \in [M-1] \\ \texttt{fee}((M-1)\Delta), & \text{if} \quad (M-1)\Delta \leq t \end{cases} \tag{8.17}$$

In our environment, block generation time follows the exponential distribution with rate λ.



(a) January 1, 2023

(b) December 17, 2023

Figure 8.13: Linear time-fee equation.

8.A5 Limitations of the Simplified Environment

Non-linearity of the time-fee equation: In Figure 8.13 presented in Appendix 8.A4, a linear regression model is used to represent the time-fee equation. However, the block transaction fee does not necessarily increase linearly with the block generation time. This occurs when the rate of fee increase is not constant and decreases as the block generation time increases. In such cases, using alternative regression techniques rather than a linear model can provide a time-fee equation that better fits the behavior observed in the Bitcoin mempool. In Figure 8.14a, we compare linear regression with a curve regression model given by $f(t) = 0.6414\ \texttt{ln}(t + 6.5209) - 0.8419$ to derive the time-fee equation for Bitcoin blocks mined on May 3, 2023. The coefficient of determination (R-squared) for the curve regression is 0.6243, while for the linear regression it is 0.5180. This indicates that the curve regression provides a better fit compared to the linear regression, suggesting that the fee-increase rate diminishes with longer block generation times.

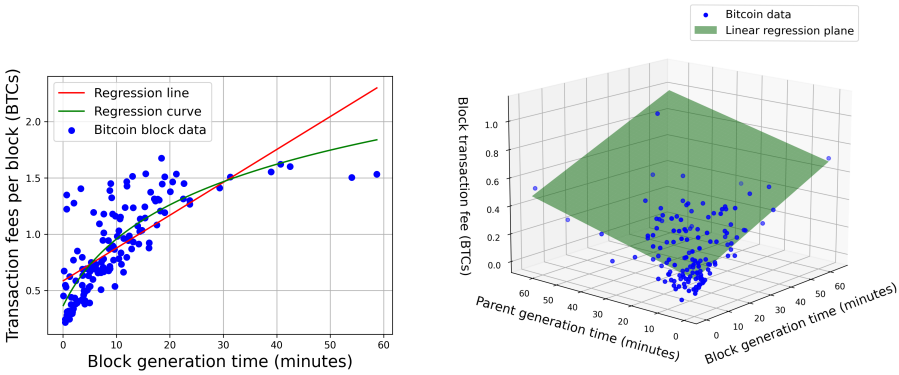
Dependence of fee rewards on the generation times of preceding blocks: Another technique that can lead to a more realistic mempool implementation is using multivariate regression instead of univariate regression. In univariate regression, the block transaction fee is estimated based solely on a single predictor, namely the block generation time. In contrast, multivariate regression allows us to account for additional predictors, such as the generation time of the parent block, in determining the block transaction fee. In Figure 8.14b, we depict the regression plane for the block transaction fee on March 18, 2024, based on two predictors: the block generation time and the parent block generation time. The resulting time-fee equation is $f(t_1, t_2) = 0.1697 + 0.0079 * t_1 + 0.0046 * t_2$, where t_1 and t_2 represent the block generation time and the parent block generation time, respectively. Note that if the parent block generation time is longer, more fee-valuable transactions will likely remain in the mempool after the parent block is mined, allowing the next block to collect these transactions. As the time-fee equation shows, the fee increase rate with respect to the block generation time is higher than the fee increase rate with respect to the parent block generation time. This implies that the block generation time is a more effective predictor of the transaction fee compared to the parent block generation time.

8.A6 A3C Implementation

8.A6.1 Objective Function

The training process aims to enable agents to identify which action maximizes the objective function at a given state s. The objective function is defined in terms of the parameters that the agent, the adversary in our case, seeks to maximize. In our implementation, this objective is to maximize the mining time-averaged profit. We discuss how the objective function can be defined both before and after the difficulty adjustment. In our implementation, at each time step t, a block is generated either by the adversary or by the honest miners. However, depending on the adversarial strategy, the block generated at time step t may not immediately be added to the canonical chain. This block could be added to the canonical chain or even be identified as orphaned at a future time step. Let $N_{\mathcal{A}}(t;\pi)$, $N_{\mathcal{H}}(t;\pi)$, and $R_{\mathcal{A}}(t;\pi)$ denote the number of adversarial blocks, the number of honest blocks, and the total reward included in the adversarial blocks, respectively, added to the canonical chain at time step t under policy π. Also, let $N_{\mathcal{T}}(t;\pi)$ denote the total number of blocks both added to and removed (orphaned) from the canonical chain at time step t under policy π. For each time step t, we have $N_{\mathcal{T}}(t;\pi) \geq N_{\mathcal{A}}(t;\pi) + N_{\mathcal{H}}(t;\pi)$. After the state transition at time step t, the environment returns the next state along with the following information to the agent: $\big(R_{\mathcal{A}}(t;\pi), N_{\mathcal{A}}(t;\pi) + N_{\mathcal{H}}(t;\pi), N_{\mathcal{T}}(t;\pi)\big)$.

We denote by $\mathbf{t}^B$ the average block generation time. We normalize the average block generation time achieved while all the miners including the adversary mine honestly to $\mathbf{t}^B_{\texttt{ideal}} = 1$. Let us ignore the mining cost since, as long as the



(a) May 3, 2023 (b) March 18, 2024

Figure 8.14: Time-fee equation.

adversary does not turn off some of its miners, its mining cost is not affected by the chosen strategy. Therefore, the time-averaged profit can be defined as follows:

$$\texttt{Profit}^{\mathcal{A}}(\pi) = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} R_{\mathcal{A}}(t;\pi)}{\mathbf{t}^{B} \cdot T} \quad . \tag{8.18}$$

Depending on the mining difficulty and the adversary's strategy, the average block generation time $\mathbf{t}^{B}$ may vary, leading to different objective functions before and after the difficulty adjustment.

8.A6.1.1 Before the Difficulty Adjustment

As discussed in Section 8.2.2.1, before a difficulty adjustment, the adversary's mining strategy does not affect the average block generation time, and thus, $\mathbf{t}^{B} = 1$. Note that while the canonical block generation time may differ, the average duration of each time step in our environment before a difficulty adjustment is determined by the average of both canonical and non-canonical block generation times, which remains unchanged. Therefore, the objective function before the difficulty adjustment can be defined as follows:

$$O^{\mathcal{A}}(\pi) = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} R_{\mathcal{A}}(t;\pi)}{T} \quad . \tag{8.19}$$

To find the policy that maximizes the optimization function above, the reward that agents receive after a transition at time step t, referred to as the *time step reward* and denoted by r_t, is set to $R_{\mathcal{A}}(t;\pi)$. During our implementation, it was found that including the mining cost in the reward function (as a constant negative reward added after each time step to account for the passage of time) and normalizing the rewards helps the agent converge faster and in a more stable manner. Our implementation uses the following step reward function:

$$r_t = \frac{R_{\mathcal{A}}(t;\pi)}{R^{\texttt{norm}}} - \texttt{cost} \ , \tag{8.20}$$

where R^{norm} denotes the total distributed reward on average per $\mathbf{t}^{B}$ units of time if all miners mine honestly, and `cost` denotes the constant cost per time step.

8.A6.1.2 After the Difficulty Adjustment

After the difficulty adjustment, the mining difficulty will be modified, resulting in a decrease in the block generation time. The average block generation time after difficulty adjustment and up to including time step T can be obtained as

follows:

$$\mathbf{t}^B = \frac{\sum_{t=0}^{T} N_{\mathcal{A}}(t;\pi) + N_{\mathcal{H}}(t;\pi)}{\sum_{t=0}^{T} N_{\mathcal{T}}(t;\pi)} \cdot \mathbf{t}^B_{\texttt{ideal}} = \frac{\sum_{t=0}^{T} N_{\mathcal{A}}(t;\pi) + N_{\mathcal{H}}(t;\pi)}{T} \quad . \tag{8.21}$$

The second equality in the equation above is achieved as $\mathbf{t}^B_{\texttt{ideal}} = 1$ and $\sum_{t=0}^{T} N_{\mathcal{T}}(t) = T$. Based on the value of $\mathbf{t}^B$ and the time-averaged profit formula outlined in Equation 8.18, we define the objective function after difficulty adjustment as follows:

$$O^{\mathcal{A}}(\pi) = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} R_{\mathcal{A}}(t;\pi)}{\sum_{t=0}^{T} N_{\mathcal{A}}(t;\pi) + N_{\mathcal{H}}(t;\pi)} \quad . \tag{8.22}$$

Since the objective function defined in equation 8.22 is non-linear, we cannot naively assign the time step reward as is done in Section 8.A6.1.1. To address this challenge, we adopt the approach introduced in [23], which converts the non-linear optimization function into a linear one. We set the time step reward as follows:

$$r_t(\rho) = R_{\mathcal{A}}(t;\pi) - \rho\big(N_{\mathcal{A}}(t;\pi) + N_{\mathcal{H}}(t;\pi)\big) \quad , \tag{8.23}$$

where ρ represents the time-averaged profit the adversary earns by following the optimal strategy, i.e., $\rho = O^{\mathcal{A}}(\pi^*)$, where π^* denotes the optimal strategy. At the start of training, the actual value of ρ is unknown. Therefore, the training agents initialize with $\rho = O^{\mathcal{A}}(\pi^{\mathcal{H}})$, where $\pi^{\mathcal{H}}$ denotes the honest strategy. Certain test agents are tasked with measuring the time-averaged profit under the policy trained by the training agents at specific time steps during the training process. If this profit exceeds the highest time-averaged profit achieved so far, they prompt the training agents to update ρ to the new highest time-averaged profit. For the detailed analysis of the non-linear objective function conversion, readers are referred to [23]. We briefly provide the intuition behind setting the time step reward as defined in equation 8.23 in the following. The value function of the initializing state s_0 under the time step reward introduced in equation 8.23 can be obtained as follows:

$$V^{\mathcal{A}}(s_0;\pi,\rho) = \sum_{t=0}^{\infty} \gamma^t R_{\mathcal{A}}(t;\pi) - \rho \sum_{t=0}^{\infty} \gamma^t \big(N_{\mathcal{A}}(t;\pi) + N_{\mathcal{H}}(t;\pi)\big) \quad . \tag{8.24}$$

Let ρ_1 be the time-averaged profit achieved under policy π_1, i.e., $\rho_1 = O^{\mathcal{A}}(\pi_1)$. If, for a given policy π_2 and as $\gamma \to 1$, we have $V^{\mathcal{A}}(s_0;\pi_2,\rho_1) > 0$, then we can conclude π_2 dominates π_1. This is because $V^{\mathcal{A}}(s_0;\pi_2,\rho_1) > 0$ implies that:

$$\frac{\sum_{t=0}^{\infty} \gamma^t R_{\mathcal{A}}(t;\pi_2)}{\sum_{t=0}^{\infty} \gamma^t \big(N_{\mathcal{A}}(t;\pi_2) + N_{\mathcal{H}}(t;\pi_2)\big)} > \rho_1 \quad .$$

According to equation 8.22 and considering $\gamma \to 1$, the inequality above results in $O^{\mathcal{A}}(\pi_2) > O^{\mathcal{A}}(\pi_1)$, which indicates that following strategy π_2 is more profitable than π_1. Therefore, if the training agents under step rewards $r_t(\rho_1)$ are properly trained, the resulting policy will achieve a time-averaged profit ρ_2, where $\rho_2 \geq \rho_1$. If, in the next step, the agents are trained based on the step reward $r_t(\rho_2)$, the time-averaged profit of the resulting policy will be ρ_3 with $\rho_3 \geq \rho_2$. By continuing this process, the trained policy will gradually converge to the optimal policy that yields the highest time-averaged profit.

Another critical point is that the average time step duration after difficulty adjustment needs recalibration in the environment. In our implementation, the duration of each time step follows an exponential distribution with rate $\frac{1}{t^B}$. The duration of each time step is of great importance, as it affects the amount by which transaction weight increases in the mempool. Before the difficulty adjustment, t^B is set to 10 minutes. However, after the difficulty adjustment, the value of block generation time t^B needs to be adjusted according to the policy followed by the adversary. To calculate the block generation time after difficulty adjustment under policy π, we need to keep track of the ratio of the number of canonical blocks to the total number of blocks mined under policy π. To this end, the neural network outputs another value that aims to estimate the following sum:

$$\sum_{t=0}^{T} \gamma^t \big(N_{\mathcal{A}}(t;\pi) + N_{\mathcal{H}}(t;\pi)\big) \ . \tag{8.25}$$

The neural network is trained to estimate this sum based on the values of $N_{\mathcal{A}}(t;\pi)$ and $N_{\mathcal{H}}(t;\pi)$ at each time step. With access to the sum of the number of canonical blocks and equation 8.21, the environment can calculate the block generation time. Note that the policy is only trained to maximize the optimization function defined in equation 8.22, and this sum is not used in training the policy.

8.A6.2 Implementation Details

Our A3C-based tool is adapted from the code provided in [19]. Our implementation consists of two linear layers, each with 256 nodes, followed by an LSTM layer that processes the output from the last linear layer. The model uses a learning rate of 10^{-6}, a discount factor of 0.99, and an entropy coefficient of 0.01. We employ 30 training agents to train the neural network and 2 testing agents to evaluate the results. During the training process, information related to the three best checkpoints is stored, and results are presented after running those checkpoints for 10^6 steps.

8.A6.3 Implementation of the Undercutting Attack

To analyze the undercutting attack, the possible action set for the A3C agents is limited to the following actions:

- If the adversary mines a new block on top of the longest chain, it publishes the block for the other miners.
- Whenever the tip of the longest chain, denoted by block B_1, is not an adversarial block, the adversary attempts to undercut block B_1. For a period of t seconds, determined by the training process, the adversary mines on the parent of block B_1. If no block is mined during this time, the adversary abandons the undercutting attempt and resumes mining on top of block B_1.
- If, during the undercutting of block B_1, the next block B_2 is mined by honest miners on top of B_1, the adversary stops undercutting block B_1 and begins undercutting block B_2.
- If, while undercutting block B_1, the adversary successfully mines a competing block $B_{\mathcal{A}}$, it publishes block $B_{\mathcal{A}}$. If the next block B_2 is mined on top of B_1, the undercutting attempt fails, and the adversary tries to undercut block B_2. If block B_2 is mined on top of block $B_{\mathcal{A}}$, the attack succeeds.

Note that this set of actions does not result in obtaining the optimal strategy and is only designed to measure the profitability of single-block undercutting under different mempool patterns.

8.A7 Selfish Mining After Difficulty Adjustment

In Figure 8.15, the percentage increase in time-averaged profit of selfish mining after a difficulty adjustment is depicted as a function of the adversarial communication capability. This figure is depicted based on the following assumptions: i) the protocol reward is set to 0 BTC, ii) the mempool pattern observed on December 18, 2023, between 23:00 and 24:00 CET, has occurred, iii) the adversary's mining share is equal to $\frac{1}{3}$. As shown in Figure 8.15, under the mempool pattern observed on December 18, 2023, difficulty-adjusted selfish mining is more profitable in the fixed reward model compared to the volatile reward model. This suggests that, under selfish mining, the rate of increase in collected transaction fees is lower than the rate of increase in the adversarial block ratio. One of the reasons contributing to the higher profitability of selfish mining in the fixed reward model is the concavity of the time-fee function during this period. Figure 8.16 illustrates the time-fee function obtained under the mempool pattern from December 18, 2023, between 23:00 and 24:00 CET. To generate Figure 8.16, we simulated 100,000 blocks under this mempool pattern, collecting their generation times and fees. As depicted in Figure 8.16, the corresponding time-fee function for this period is concave.

To better understand the effect of the concavity of the time-fee function on selfish mining profitability, we review the following theorem.

Theorem 8.1. If the time-fee function is strictly concave, the average fee of a canonical block under honest mining will be greater than the average fee of a canonical block under difficulty-adjusted selfish mining.

Proof. Let $f(t)$ denote the fee of a block with a block generation time of t, where $f(t)$ is strictly concave. Let λ and λ' denote the total block generation rates under honest mining and difficulty-adjusted selfish mining, respectively. Note that the block generation time follows an exponential distribution. The canonical block generation rate under honest mining is also equal to λ. Therefore, the average canonical block fee under honest mining, denoted by $r_{\mathcal{H}}$, can be obtained as follows:

$$r_{\mathcal{H}} = \int_0^\infty f(t)\lambda e^{-\lambda t}\, dt \ . \tag{8.26}$$

We divide canonical blocks mined under difficulty-adjusted selfish mining into groups $\{G_1, G_2, G_3, \cdots\}$. A canonical block B belongs to group G_i for $i \in \{1, 2, 3, \cdots\}$ if the number of orphaned blocks mined between the parent of block B and block B itself is $i - 1$. The average canonical block generation rate of blocks belonging to group G_i is equal to $\frac{\lambda'}{i}$. Let a_i denote the ratio of the number of blocks belonging to group G_i to the total number of canonical blocks, where $\sum_{i=1}^{\infty} a_i = 1$. As the average canonical block generation rate

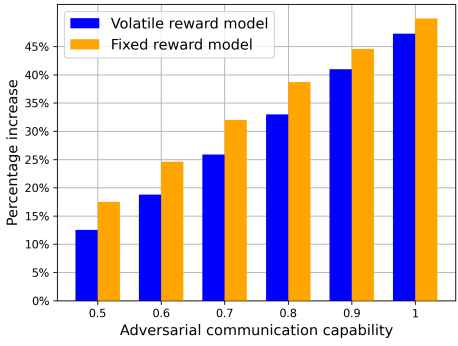


Figure 8.15: The profit percentage increase of selfish mining after difficulty adjustment.

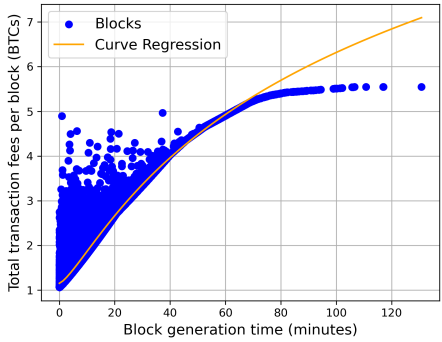


Figure 8.16: The concave time-fee function.

under difficulty-adjusted selfish mining is also equal to λ, we have:

$$\frac{1}{\lambda} = \sum_{i=1}^{\infty} a_i \int_0^{\infty} t' \frac{\lambda'}{i} e^{-\frac{\lambda'}{i}t'} \, dt' \ . \tag{8.27}$$

By calculating the result of integral, we have:

$$\frac{1}{\lambda} = \frac{\sum_{i=1}^{\infty} a_i i}{\lambda'} \Longrightarrow \frac{\lambda'}{\lambda} = \sum_{i=1}^{\infty} a_i i \ . \tag{8.28}$$

The average fee of a canonical block under difficulty-adjusted selfish mining, denoted by $r_{\mathcal{S}}$, can be obtained as follows:

$$r_{\mathcal{S}} = \sum_{i=1}^{\infty} a_i \int_0^{\infty} f(t') \frac{\lambda'}{i} e^{-\frac{\lambda'}{i}t'} \, dt' \ . \tag{8.29}$$

By making a variable substitution of form $\lambda t = \frac{\lambda'}{i} t'$, we obtain:

$$r_{\mathcal{S}} = \sum_{i=1}^{\infty} a_i \int_0^{\infty} f(\frac{\lambda}{\lambda'} it) \lambda e^{-\lambda t} \, dt \ . \tag{8.30}$$

As function $f(t)$ is strictly concave, we have:

$$r_{\mathcal{S}} < \int_0^{\infty} f(\frac{\lambda}{\lambda'} (\sum_{i=1}^{\infty} a_i i) t) \lambda e^{-\lambda t} \, dt \ . \tag{8.31}$$

Using the result obtained in equation 8.28, we can write the inequality above as follows:

$$r_{\mathcal{S}} < \int_0^{\infty} f(t) \lambda e^{-\lambda t} \, dt \ . \tag{8.32}$$

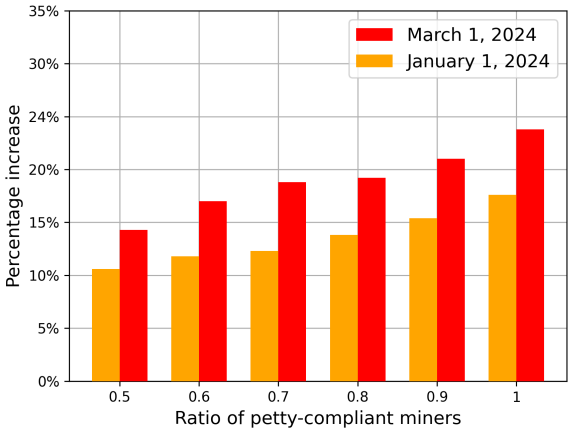


Figure 8.17: The percentage increase in time-averaged profit.

The right hand side in inequality above is equal to $r_{\mathcal{H}}$. Therefore, we obtain $r_{\mathcal{S}} < r_{\mathcal{H}}$. □

According to Theorem 8.1, although the average canonical block generation rate is the same under both honest mining and difficulty-adjusted selfish mining, the total transaction fees distributed among canonical blocks under difficulty-adjusted selfish mining decrease due to the concavity of the time-fee function. This results in a lower profit percentage increase in the volatile model compared to honest mining.

8.A8 Selfish Mining Before Difficulty Adjustment: Supplementary

Figure 8.9 shows the percentage increase in the adversary's time-averaged profit before a difficulty adjustment resulting from conducting selfish mining. This figure is depicted based on different ratios of petty-compliant miners, over the mempool patterns observed on two different days: January 1, 2024, and March 1, 2024, both between 8:00 and 9:00 AM CET. The figure is generated using the following configuration parameters: i) the protocol reward is set to 0 BTC, and ii) the adversary's mining share and communication capability are set to $\frac{1}{3}$ and 50%, respectively.

The greater percentage increase observed under the mempool pattern of March 1, 2024, can be attributed to a higher normalized rate of growth in fee rewards during this period. Specifically, the ratio of the average fee reward for a block generated in 20 minutes to that for a block generated in 10 minutes is 1.29 under the mempool pattern of January 1, 2024, and 1.4 under that of March 1, 2024. This indicates that if the adversary succeeds in orphaning a preceding block and thereby extends the block generation time for its own block, it can accumulate a relatively higher fee reward under the mempool pattern of March 1, 2024.

Chapter 9

Deep Selfish Proposing in Longest-Chain Proof-of-Stake Protocols

Publication Data

Roozbeh Sarenche, Svetla Nikova, and Bart Preneel, "Deep Selfish Proposing in Longest-Chain Proof-of-Stake Protocols," in *International Conference on Financial Cryptography and Data Security (FC)*, volume 14744, pages 24–40, Springer, 2024 (https://eprint.iacr.org/2024/622).

Contributions

I was the main author involved in Conceptualization, Methodology, Writing Original Draft, Software, and Visualization.

Abstract. It has been shown that the selfish mining attack enables a miner to achieve an unfair relative revenue, posing a threat to the progress of longest-chain blockchains. Although selfish mining is a well-studied attack in the context of Proof-of-Work blockchains, its impact on the longest-chain Proof-of-Stake (LC-PoS) protocols needs yet to be addressed. This paper involves both theoretical and implementation-based approaches to analyze selfish proposing[1] attack in the LC-PoS protocols. We discuss how factors such as the nothing-at-stake phenomenon and the proposer predictability in PoS protocols can make the selfish proposing attack in LC-PoS protocols more destructive compared to selfish mining in PoW. In the first part of the paper, we use combinatorial tools to theoretically assess the selfish proposer's block ratio in simplistic LC-PoS environments and under simplified network connection. However, these theoretical tools or classical MDP-based approaches cannot be applied to analyze the selfish proposing attack in real-world and more complicated LC-PoS environments. To overcome this issue, in the second part of the paper, we employ deep reinforcement learning techniques to find the near-optimal strategy of selfish proposing in more sophisticated protocols. The tool implemented in the paper can help us analyze the selfish proposing attack across diverse blockchain protocols with different reward mechanisms, predictability levels, and network conditions.

9.1 Introduction

In Proof-of-Stake (PoS) blockchains, proposers—namely, users who have deposited a specific amount of stake in the blockchain ledger—are responsible for proposing new blocks and extending the blockchain ledger [4]. Once a proposer is selected to propose a block, it needs to follow a pre-determined fork choice rule to select a chain on top of which the new block will be proposed. Fork choice rules are essential for blockchains to achieve safety and liveness [21]. Bitcoin [20] uses the longest-chain fork choice rule, where miners choose the longest chain to mine on top of it. Although the longest-chain rule was originally designed for Proof-of-Work (PoW), some PoS-based blockchain protocols have also implemented the longest-chain fork choice rule within their consensus layers, such as the Ouroboros protocol in Cardano [10, 17], the Emmy protocol in Tezos [15], and Snow White [9]. In PoW, the longest chain refers to the chain with the most amount of work. However, in PoS, since there is no mining process, the longest chain refers to the chain that is literally the longest.

One of the most important attacks that can threaten longest-chain-based blockchain protocols is the selfish mining attack presented by Eyal and Sirer [12].

[1] As there is no mining process in PoS blockchains, we refer to this attack as "selfish proposing".

The authors of this paper have shown that a malicious miner in Bitcoin can increase his payoff by deviating from mining honestly. In the selfish mining attack, once the attacker mines a new block, he keeps his block secret rather than immediately publishing it to the other mining nodes. By keeping his blocks secret, the attacker can cause some part of the honest mining power to get wasted since the honest nodes continue mining on top of the public chain which is shorter than the attacker's secret chain. Due to the selfish mining attack, some of the honest blocks get orphaned, i.e., get excluded from the main chain. This results in an increase in the ratio of the number of attacker's blocks added to the main chain to the total number of main-chain blocks, which we refer to as the attacker's "block ratio". The selfish mining attack is applicable to any blockchain that is designed based on the longest-chain fork choice rule. Thus, not only the PoW-based blockchains but also PoS-based blockchains that use the longest-chain paradigm face the threat of selfish mining. Note that since no mining process is involved in PoS-based blockchains, we use the term "selfish proposing" rather than selfish mining to describe this attack in the PoS context.

There exist several reasons that make selfish proposing in PoS even more destructive compared to selfish mining in PoW. The first reason is the presence of the nothing-at-stake phenomenon in PoS-based blockchains. This phenomenon implies that since there is no mining process in PoS protocols, generating multiple blocks could be wasteless for proposers [7]. Therefore, a PoS proposer has the flexibility to modify the content of an already generated block as long as the block is not yet published. However, this is not the case in PoW-based blockchains. A PoW miner should specify the block content prior to the start of the mining process, and the block content cannot be changed once it is mined. The block content, in both PoW and PoS, includes a reference to its parent block. In PoS-based blockchains, the nothing-at-stake phenomenon allows a selfish proposer to postpone specifying the block's parent until just before the block needs to be published. This can provide a selfish proposer in PoS-based blockchains with a set of new actions to fortify the selfish proposing attack [13].

Another bottleneck of PoS protocols that leads to a higher selfish proposing profit is the proposer predictability. In PoS protocols, there exists a lottery mechanism that specifies the block proposer(s) for each slot. To select proposers in a random manner, the lottery mechanism takes a pseudorandom seed as one of its inputs. This seed, in most PoS-based blockchains, is extracted from the blockchain ledger information. To ensure consistency and liveness, the pseudorandom seed for each slot needs to be determined when we are sufficiently far from that specific slot [3]. Knowing these pseudorandom seeds can help validators predict the block proposers in future slots, which is impossible in PoW. Based on how the PoS protocol is designed, the validators can have different levels of predictability. In some protocols, validators can only predict the slots in which they themselves are selected as proposers, while in other protocols,

the validators can predict the block proposers for all the upcoming slots. In both of these protocols, a selfish proposer can use information regarding the future proposers to improve his selfish proposing strategy. During a selfish mining attack in PoW, some of the attacker's blocks may get orphaned. In PoS protocols having knowledge about future slot proposers can help a selfish proposer reduce the risk of losing his blocks during the attack and increase his block ratio.

In PoW blockchains, tools such as the Markov chain and the Markov Decision Process (MDP) can be used to obtain the selfish miner's optimal strategies [22, 24]. These tools are useful for analyzing environments with a limited set of actions and states. However, to consider the effect of the nothing-at-stake phenomenon and predictability on the selfish proposer's strategy in longest-chain PoS blockchains, it is necessary to handle larger sets of actions and states. Therefore, the classical MDP-based tools cannot directly be used to analyze the selfish proposing attack in longest-chain PoS protocols. One approach to assess the environments with huge sets of actions and states is to use deep reinforcement learning techniques such as deep Q-learning. These techniques not only enable us to handle larger sets of actions and states but also facilitate the study of the selfish proposing profitability under more realistic and complicated environments. There exist research papers such as [5, 6] that have used deep reinforcement learning techniques to analyze selfish mining in PoW protocols. However, developing a tool for analyzing the selfish proposing attack in longest-chain PoS has yet to be addressed.

In this paper, we assess the selfish proposing attack in the longest-chain PoS (LC-PoS) blockchains. In Appendix 9.A2, we introduce our model which is built upon the model presented in [7]. In our paper, we extend and modify the PoS blockchain model and definitions presented in [7] to propose a realistic model that fits better to the PoS protocols currently in use. In Appendix 9.A3, we define the terms "block ratio" and "time-averaged profit" and discuss the impact of the selfish proposing attack on the attacker's profitability in LC-PoS protocols, comparing it to the profitability of selfish mining in PoW. In fact, we explain how an increase in the attacker's block ratio due to selfish proposing can lead to a rise in the attacker's time-averaged profit. In the main body of the paper, we present:

Nothing-at-stake selfish proposing attack in LC-PoS In Section 9.2, we generalize the nothing-at-stake selfish proposing attack introduced in [13] by taking into account a set of new actions that can lead to a more profitable strategy. Besides, we prove interesting lower bounds on the minimum amount of stake share that can make the selfish proposing attack more profitable than honest proposing in LC-PoS protocols with perfect randomness.

Predictable selfish proposing attack in full-predictable LC-PoS In Section 9.3, we use combinatorial tools to theoretically assess the selfish proposing block ratio under multiple strategies in full-predictable LC-PoS protocols. Moreover, we present an optimal selfish proposing strategy for full-predictable LC-PoS protocols.

Deep Q-learning tool to analyze the selfish proposing attack In Section 9.4, we present a deep Q-learning tool to analyze the selfish proposing attack in realistic and more complicated LC-PoS environments such as semi-predictable protocols. We demonstrate how this tool can assist in discovering the near-optimal selfish proposing strategy for selfish proposers with varying stake shares and communication capabilities under different LC-PoS environments.

In the main text, terms such as "honest strategy", "communication capability", and "block ratio" are frequently used, which are defined in Definitions 9.A3.1, 9.A3.2, and 9.A3.3 presented in Appendix 9.A3, respectively. Readers are encouraged to review these definitions before delving into the main text.

9.2 Nothing-at-Stake Selfish Proposing in LC-PoS Protocols

A fundamental difference between LC-PoS protocols and PoW protocols is the fact that generating a new valid block in LC-PoS protocol is effortless. This phenomenon which is known as nothing-at-stake can provide an attacker in LC-PoS protocols with a set of new actions and strategies that are impossible in PoW protocols. The authors in [13] have introduced the nothing-at-stake selfish proposing attack. However, the introduced attack is limited to an attacker with communication capability equal to 0 and can only benefit from a small set of new actions made possible by the nothing-at-stake phenomenon. In this paper, we generalize nothing-at-stake selfish proposing by considering a set of new actions that can lead to an increase in selfish proposing profitability.

Note that throughout this paper, we assume that a block proposer is either honest or adversarial. The honest proposers, denoted by $\mathcal{H}$, always follow the honest strategy. The adversarial proposers are under the control of an attacker denoted by $\mathcal{A}$ and can deviate from the honest strategy. We denote by α and η the attacker's stake share and communication capability, respectively.

9.2.1 Intuition Behind Nothing-at-Stake Selfish Proposing

We first explain how nothing-at-stake can lead to a more profitable selfish proposing attack. In PoW protocols, a miner should specify the content of a block including a reference to its parent prior to the start of the mining process. After a block is successfully mined in PoW protocols, modifying its parent block requires a significant amount of computational work since the miner needs to mine an entirely new block to include a different parent. However, in PoS protocols, a proposer can still change the content of a block after it is proposed and before it is published. In fact, once a proposer is eligible to propose a block in a PoS protocol, he has the freedom to change the parent of the block as long as it is not published without any further effort.

To get an intuition that how the nothing-at-stake phenomenon can fortify selfish proposing, consider the picture depicted in Figure 9.1. In this figure, which is based on the single-proposer model introduced in Definition 9.A2.10, the block proposed by proposer P at time slot t is denoted by B_t^P. The block proposer P can be either honest or adversarial, namely $P \in \{\mathcal{H}, \mathcal{A}\}$. As can be seen in Figure 9.1, the honest chain has a 2-block lead over the adversarial chain. We assume that none of the adversarial blocks in Figure 9.1 is published. In a similar situation in PoW protocols, the attacker should decide whether to give up on his fork and continue mining on top of the honest chain (block $B_9^{\mathcal{H}}$) or to continue mining on top of his secret chain. On the one hand, if the attacker chooses to give up, then his 4 adversarial blocks get orphaned. On the other hand, if the attacker chooses to continue mining on top of his secret chain that is 2 blocks behind, he has a relatively low chance of catching up the public chain and may risk losing his future blocks. In LC-PoS protocols, however, besides these two actions, the attacker can perform other reasonable actions. For the scenario depicted in Figure 9.1, we explain two actions that a selfish proposer can only take in LC-PoS protocols. Assume the attacker needs to decide on an action after that the honest block $B_9^{\mathcal{H}}$ is proposed at time slot $t = 9$. As the first reasonable action, the attacker can give up on block $B_1^{\mathcal{A}}$, change the parent of $B_4^{\mathcal{A}}$ to $B_3^{\mathcal{H}}$, and generate a new fork that includes 3 adversarial blocks. In this case, the length of both adversarial and honest chains gets equal to 3, and the attacker has a higher chance to orphan the honest chain. As the second reasonable action, the attacker can give up on blocks $B_1^{\mathcal{A}}$ and $B_4^{\mathcal{A}}$, change the parent of $B_7^{\mathcal{A}}$ to $B_6^{\mathcal{H}}$, and generate a new fork that includes 2 adversarial blocks. In this case, the adversarial chain has a 1-block lead over the honest chain. These two actions are examples of a broader set of actions, which we refer to as "`jump`".

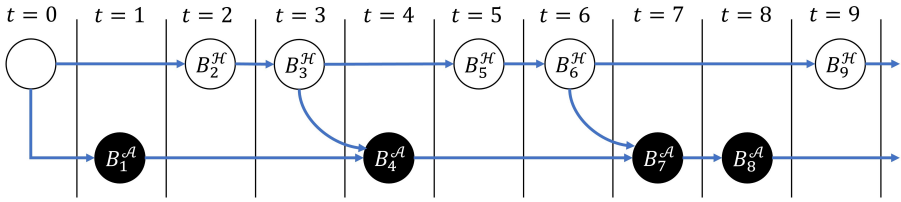


Figure 9.1: Nothing-at-stake selfish proposing

9.2.2 Set of Actions for Nothing-at-Stake Selfish Proposing

In this section, we introduce the set of actions for selfish proposing in LC-PoS protocols. We assume that the PoS protocol is accompanied by a slashing mechanism, i.e., a malicious proposer who publishes two or more contradicting blocks can get slashed by the other proposers. Therefore, a selfish proposer can publish at most one block in each slot. Note that when discussing the attacker's actions, it is important to distinguish between the terms "proposing" and "publishing" blocks. The term that the attacker proposes a block at time slot t indicates that the attacker is eligible to create a block at time slot t; however, the content of the block may not be finalized yet. The term that the attacker publishes a block at time slot t indicates that the content of the block proposed at time slot t is finalized and honest proposers are aware of that. Regarding honest proposers, the terms "proposing" and "publishing" can be used interchangeably since an honest proposer publishes its block immediately after it is proposed.

A brief overview of selfish mining in PoW protocols is presented in Appendix 9.A1.1. In our model, we assume that the attacker works on a secret chain while the honest proposers work on a single public chain. Let t_0 be the time slot up to which both the attacker's chain and the public chain share the same subchain. This indicates that after time slot t_0, the honest and adversarial chains have diverged. We refer to the attacker's chain (honest chain) proposed after t_0 as the adversarial fork (honest fork). Let $l^{\mathcal{A}}$ and $l^{\mathcal{H}}$ denote the length of the adversarial fork and the length of the honest fork, respectively. Additionally, let t_i for $i \geq 1$ represent the time slot at which the ith block is proposed in the honest fork. The actions are a combination of two subactions and have the format $(\texttt{subaction}_1, \texttt{subaction}_2)$, where

$$\begin{aligned} &\texttt{subaction}_1 \in \left\{ \texttt{jump}_i \;\middle|\; i \in \{0, 1, 2, \cdots, l^{\mathcal{H}}\} \right\} , \\ &\texttt{subaction}_2 \in \{\texttt{override}, \texttt{match}, \texttt{wait}\} . \end{aligned} \tag{9.1}$$

$\texttt{jump}_i$: The subaction represents that the attacker specifies the starting point of the fork. If $i = 0$, the attacker continues working on top of the current fork. For $i \geq 1$, the attacker gives up on the blocks in the adversarial fork proposed before including time slot t_i, adopts the honest fork up to including time slot t_i, and generates a new fork on top of the honest block proposed at time slot t_i. In this case, the new adversarial (honest) fork includes all the adversarial (honest) blocks proposed after the time slot t_i. Note that:

- For $i = 0$, the subaction is always feasible.
- For $i \geq 1$, the subaction is feasible if all the adversarial blocks proposed after t_i are unpublished.

If no adversarial block is proposed after the last block in the honest fork, one can consider $\texttt{jump}_{l^{\mathcal{H}}}$ as the action `adopt` employed in PoW blockchains, where the attacker gives up on his private chain and continues mining on top of the longest public chain. Once the starting point of the fork is specified using action $\texttt{jump}_i$, the attacker decides on which of the following subactions to perform on the new fork. Note that values of $l^{\mathcal{H}}$ and $l^{\mathcal{A}}$ may get updated after applying action $\texttt{jump}_i$.

`override`: The subaction represents that the attacker publishes his secret fork whose length is one block longer than the honest fork. This subaction is feasible if $l^{\mathcal{A}} > l^{\mathcal{H}}$.

`match`: The subaction represents that the attacker publishes his secret fork whose length is equal to the honest fork immediately after an honest block is published. This subaction is feasible if $l^{\mathcal{A}} \geq l^{\mathcal{H}}$ and $l^{\mathcal{A}} \neq 0$.

`wait`: The subaction represents that the attacker continues proposing new blocks in the specified fork. This subaction is always feasible.

9.2.3 Nothing-at-Stake Selfish Proposing with Perfect Randomness

In this section, we introduce a selfish proposing strategy suitable for longest-chain PoS protocols with perfect randomness, denoted by $\pi^{\text{S-PR}}$. Here, the perfect randomness implies that the random seed of each time slot is revealed at the start of the time slot, and consequently, no proposer can predict the slot seed prior to the start of the slot. This indicates that similar to selfish mining in PoW, a selfish proposer in an LC-PoS protocol with perfect randomness does not enjoy predictability. Thus, the only difference between selfish mining in PoW and selfish proposing in LC-PoS with perfect randomness is the nothing-at-stake phenomenon. The strategy $\pi^{\text{S-PR}}$ and a comprehensive theoretical analysis of the block ratio gained by following this strategy are presented in Appendix 9.A4. Note that strategy $\pi^{\text{S-PR}}$ is not the optimal strategy that a

selfish proposer can follow in an LC-PoS protocol with perfect randomness. However, we prove in Appendix 9.A4 that for some range of the stake share, the introduced strategy can dominate the optimal selfish mining strategy in PoW [22] and the nothing-at-stake selfish mining strategy introduced in [13]. Later in Section 9.4.2, we introduce a deep Q learning-based method to obtain the near-optimal selfish proposing strategy in an LC-PoS protocol with perfect randomness.

In the following, we only review the amount of profitable stake share threshold (introduced in Definition 9.A3.4) for a selfish proposer following the strategy $\pi^{\text{S-PR}}$. In PoW protocols, there is a famous bound that indicates the profitable mining share threshold of an attacker with communication capability $\eta = 0.5$ is equal to 0.25 [12, 22]. According to the analysis presented in Appendix 9.A4, the profitable stake share threshold of a selfish proposer with communication capability $\eta = 0.5$ in an LC-PoS protocol with perfect randomness is less than 0.24198. This indicates that a selfish proposer whose stake share is greater than or equal to 0.24198 can increase his block ratio by deviating from the honest strategy in an LC-PoS protocol with perfect randomness.

We also show in Appendix 9.A4 that the profitable stake share threshold of a selfish proposer with communication capability $\eta = 0$ in an LC-PoS protocol with perfect randomness is less than 0.323606. This threshold is lower than the profitable stake (mining) share threshold achieved by following the optimal PoW strategy and the nothing-at-stake selfish mining strategy introduced in [13], which are almost equal to 0.329 and 0.324718, respectively.

9.3 Selfish Proposing in Predictable LC-PoS Protocols

In this section, we aim to assess the effect of proposer predictability in fortifying the selfish proposing attack. Note that a predictable PoS protocol can be either semi-predictable (Definition 9.A2.13) or full-predictable (Definition 9.A2.14). We discuss how malicious proposers can use information regarding the upcoming block proposers in both semi and full-predictable environments to apply a successful selfish proposing attack with a lower risk of losing their blocks.

In a full-predictable protocol, all the proposers including the attacker can predict the proposing sequence, the definition of which is presented in Definition 9.A2.15. Being aware of the proposing sequence in advance can help the attacker modify his strategy to increase the selfish proposing block ratio. Consider a simple example in a full-predictable PoS protocol, where the attacker is eligible to propose a block, e.g., $B^{\mathcal{A}}$, at time slot $t = 1$. The attacker should decide whether he wants to publish block $B^{\mathcal{A}}$ immediately or keep it secret in the hope

of orphaning some honest blocks. Due to the full predictability, the attacker can predict the block proposers of upcoming slots $t > 1$ at time slot $t = 1$. Assume the scenario in which the attacker is lucky and is responsible for proposing another block at time slot $t = 2$. In this scenario, since the attacker is sure that he can use his two consecutive blocks of slots $t = 1$ and $t = 2$ to orphan at least one honest block, he would keep block $B^{\mathcal{A}}$ secret. As the second scenario, assume the case in which the attacker is unlucky as he is not among the block proposers in the near subsequent slots. In this case, especially if the attacker does not enjoy a high amount of communication capability, the attacker would publish block $B^{\mathcal{A}}$ since he knows if he keeps the block secret, the block would get orphaned with a high probability.

In a semi-predictable protocol, although the attacker cannot predict a full list of upcoming block proposers, he is still capable of predicting those time slots in which he is responsible for proposing a block. The distance between these adversarial time slots can help the attacker to increase the selfish proposing success rate. Consider the same example where the attacker is eligible to propose a block, e.g., $B^{\mathcal{A}}$, at time slot t. Assume the scenario in which the attacker knows that he can propose his next block at time slot $t + \delta$, where δ is relatively small. In this scenario, the attacker may keep the block $B^{\mathcal{A}}$ secret due to the low probability of honest proposers proposing a block within the duration of δ slots. However, in the scenario where the attacker knows he has no chance to propose a block for a long period after time slot t, he would publish block $B^{\mathcal{A}}$ immediately.

9.3.1 Selfish Proposing in Full-Predictable LC-PoS Protocols

This section presents theoretical methods to analyze selfish proposing in full-predictable LC-PoS protocols for an attacker with communication capability $\eta = 0$. In this section, we assume that the underlying LC-PoS protocol is an infinite-full-predictable protocol, where the attacker can predict block proposers for an infinite number of future time slots in advance. The authors in [7] presented a selfish proposing strategy for a selfish proposer with communication capability $\eta = 0$ in a full-predictable environment. We denote this strategy by $\pi^{\mathtt{SP1}}$. Despite the introduction of strategy $\pi^{\mathtt{SP1}}$, the authors did not provide any theoretical analysis for this strategy in [7]. In this section, we first introduce strategy $\pi^{\mathtt{SP1}}$ and then calculate the attacker's block ratio under this strategy. Then, we introduce another strategy denoted by $\pi^{\mathtt{SP2}}$, which we prove can dominate strategy $\pi^{\mathtt{SP1}}$. Finally, we show that none of the strategies $\pi^{\mathtt{SP1}}$ and $\pi^{\mathtt{SP2}}$ are optimal and present an optimal strategy for a selfish proposer with communication capability $\eta = 0$ in full-predictable LC-PoS protocols. To present the selfish proposing strategies, we use the concept of the "longest

dominant chain" defined in Definition 9.A2.18. Note that $\texttt{LDC}(t)$ and $L^{\texttt{LDC}}(t)$ denote the longest dominant chain of time slot t and its corresponding length. For presenting the following strategies, we assume that the LC-PoS protocol is compatible with the single-proposer model. Strategy $\pi^{\texttt{SP1}}$ is defined as follows:

(Strategy $\pi^{\texttt{SP1}}$). At each time slot t:

- If $L^{\texttt{LDC}}(t) = 0$, the attacker does not fork the honest chain and moves to the next time slot.
- If $L^{\texttt{LDC}}(t) > 0$, where $\texttt{LDC}(t)$ ends at time slot t', the attacker forks the honest chain and keeps the adversarial chain secret within the time slot interval $[t, t']$. At the end of time slot t', the attacker publishes $\texttt{LDC}(t)$ and orphans $L^{\texttt{LDC}}(t) - 1$ honest blocks proposed within the interval $[t, t']$.

Theorem 9.1. Let α denote the stake share of an attacker $\mathcal{A}$. The block ratio of attacker $\mathcal{A}$ under strategy $\pi^{\texttt{SP1}}$ can be obtained as follows:

$$\texttt{BlkRatio}_{\mathcal{A}}(\pi^{\texttt{SP1}}) = \frac{\alpha}{1 - 2\alpha^2} \tag{9.2}$$

The proof of Theorem 9.1 is presented in Appendix 9.A5.

We introduce another strategy denoted by $\pi^{\texttt{SP2}}$, which can dominate startegy $\pi^{\texttt{SP1}}$.

(Strategy $\pi^{\texttt{SP2}}$). At each time slot t:

- If the slot proposer is honest, the attacker does not fork the honest chain and moves to the next time slot.
- If the slot proposer is the attacker himself, where $\texttt{LDC}(t)$ ends at time slot t', the attacker forks the honest chain and keeps the adversarial chain secret within the time slot interval $[t, t']$. At the end of time slot t', the attacker publishes $\texttt{LDC}(t)$ and orphans $L^{\texttt{LDC}}(t) - 1$ honest blocks proposed within the interval $[t, t']$.

Theorem 9.2. Let α denote the stake share of an attacker $\mathcal{A}$. The block ratio of attacker $\mathcal{A}$ under strategy $\pi^{\texttt{SP2}}$ can be obtained as follows:

$$\texttt{BlkRatio}_{\mathcal{A}}(\pi^{\texttt{SP2}}) = \frac{\alpha(1 - \alpha(1 - \alpha))}{(1 - \alpha)^2(1 + \alpha)} \tag{9.3}$$

The proof of Theorem 9.2 is presented in Appendix 9.A6.

For all $0 < \alpha < 0.5$, $\texttt{BlkRatio}_{\mathcal{A}}(\pi^{\texttt{SP2}}) > \texttt{BlkRatio}_{\mathcal{A}}(\pi^{\texttt{SP1}})$. By following one of the strategies $\pi^{\texttt{SP1}}$ or $\pi^{\texttt{SP2}}$, none of the adversarial blocks get orphaned during the selfish proposing attack, which is impossible in PoW selfish mining attack. Moreover, in both of these two strategies, the attacker always waits until he can generate the longest possible adversarial fork. By doing so, the attacker aims

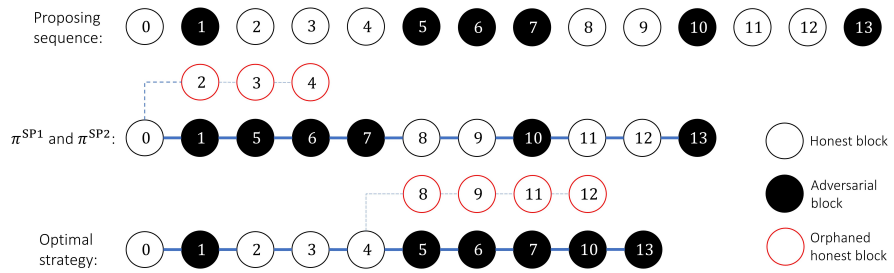


Figure 9.2: Block ratio of selfish proposing strategies in a full-predictable protocol

to maximize the number of honest blocks that get orphaned during the attack. Although strategies $\pi^{\mathtt{SP1}}$ or $\pi^{\mathtt{SP2}}$ dominate the optimal PoW selfish mining strategy, they are not the optimal strategy that an attacker with communication capability $\eta = 0$ can follow in a full-predictable LC-PoS protocol. Consider the proposing sequence depicted in Figure 9.2. Following both strategies $\pi^{\mathtt{SP1}}$ and $\pi^{\mathtt{SP2}}$, the attacker's block ratio would be equal to 6/11. However, the optimal block ratio that the attacker can achieve is equal to 6/10. In the optimal scenario, the number of adversarial blocks that get added to the main chain is the same as that of when following $\pi^{\mathtt{SP1}}$ or $\pi^{\mathtt{SP2}}$. However, by following the optimal strategy, the attacker manages to orphan a greater number of honest blocks. The example in Figure 9.2 shows that to achieve the optimal block ratio, the attacker forking the public chain at time slot t should not always wait until $\mathtt{LDC}(t)$ is generated. In fact, the attacker should sometimes publish the fork at an earlier time slot to create a better orphaning opportunity at future slots. To introduce the optimal strategy, we first define the checkpoint slot.

Definition 9.3.1 (Checkpoint slot)**.** An honest time slot t is called a checkpoint, if there is no adversarial time slot t', where $t' < t$, such that its longest dominant chain, i.e., $\mathtt{LDC}(t')$, ends at a time slot t'' with $t'' \geq t$.

An honest time slot t being a checkpoint implies that the honest block proposed in slot t cannot be orphaned by an adversarial fork initiated at a time slot earlier than t. Our checkpoint definition is similar to the Nakamoto block definition introduced in [11]. In the following, we introduce the selfish proposing optimal strategy for an attacker with communication capability $\eta = 0$ under the full-predictable model, which is denoted by $\pi^{\mathtt{O\text{-}SP}}$. Let $\mathtt{Seq}$ denote the proposing sequence within the interval $[0, \infty)$. In strategy $\pi^{\mathtt{O\text{-}SP}}$, the attacker first needs to find all the checkpoint time slots denoted by $\{t_1, t_2, \cdots\}$. Let subsequence $\mathtt{Seq}_i$ be a subset of proposing sequence $\mathtt{Seq}$ that corresponds to time slot interval $T_i = (t_i, t_{i+1})$, i.e., $\mathtt{Seq}_i = \mathtt{Seq}(T_i)$. In Appendix 9.A8, we prove that the optimal strategy of selfish proposing in each of these subsequences is independent of strategies followed in the other subsequences. Therefore, to find

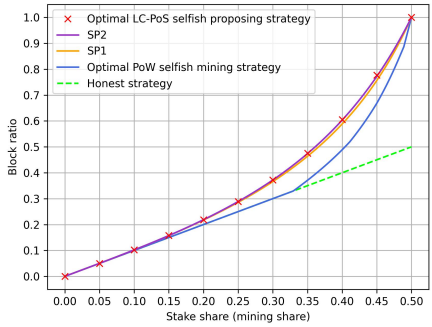


Figure 9.3: Block ratio comparison in a full-predictable protocol ($\eta = 0$)

the optimal selfish proposing strategy in the whole proposing sequence $\mathtt{Seq}$, the attacker only needs to find the optimal strategy for each individual subsequence. The optimal selfish proposing strategy for an attacker with communication capability $\eta = 0$ in a full-predictable model satisfies two properties: (1) it ensures that none of the adversarial blocks get orphaned, and (2) it maximizes the number of honest blocks that get orphaned. In an ideal scenario where $\eta = 1$, an attacker should be able to orphan one honest block per each adversarial block. However, since the attacker's communication capability is equal to 0, for each adversarial fork comprising of n adversarial blocks, the attacker can orphan at most $n - 1$ honest blocks, which is one honest block less than the ideal scenario. This indicates that when $\eta = 0$, each adversarial fork results in the inclusion of an additional honest block in the main chain compared to the ideal scenario. Therefore, to increase the number of orphaned honest blocks in each subsequence $\mathtt{Seq}_i$, the attacker should follow the strategy that minimizes the number of adversarial forks in that subsequence.

(Strategy $\pi^{\mathtt{O\text{-}SP}}$). Find all the checkpoint time slots. For each pair of two consecutive checkpoint slots t_1 and t_2, use Algorithm 2 presented in Appendix 9.A7 to find the optimal strategy.

In Appendix 9.A8, we prove that strategy $\pi^{\mathtt{O\text{-}SP}}$ is the optimal strategy that an attacker with communication capability $\eta = 0$ can follow in full-predictable LC-PoS protocols. In Figure 9.3, we compare the block ratios achieved by strategies $\pi^{\mathtt{SP1}}$, $\pi^{\mathtt{SP2}}$, and $\pi^{\mathtt{O\text{-}SP}}$ within a full-predictable LC-PoS protocol with the block ratios of the honest strategy and the optimal PoW selfish mining strategy presented in [22]. A more precise comparison among the block ratios of strategies $\pi^{\mathtt{SP1}}$, $\pi^{\mathtt{SP2}}$, and $\pi^{\mathtt{O\text{-}SP}}$ is presented in Table 9.1.

9.4 Deep Q-Learning to Analyze Selfish Proposing

Thus far, we have analyzed the selfish proposing attack in simplified LC-PoS environments such as LC-PoS protocols with perfect randomness and full-predictable LC-PoS protocols. In this section, we use Deep Q-Learning (DQL) to analyze the selfish proposing attack in more complicated scenarios, such as semi-predictable environments, and obtain the near-optimal selfish proposing strategy for all proposers with varying stake shares and communication capabilities.

9.4.1 DQL Implementation in Blockchain Environments

A brief overview of deep Q-learning is presented in Appendix 9.A1.2. In a DQL implementation that is specifically designed for blockchains, each interaction experience with the blockchain environment has the following format:

$$e_t = (s_t, a_t, s_{t+1}, r_t^{\mathcal{A}}, r_t^{\mathcal{H}}) \ , \tag{9.4}$$

where s_t is the state visited at step t, a_t is the action performed by the attacker at step t, s_{t+1} is the subsequent state resulting from taking action a_t at state s_t, and $r_t^{\mathcal{A}}$ and $r_t^{\mathcal{H}}$ are the rewards the attacker and the honest proposers received due to moving from state s_t to state s_{t+1} under action a_t, respectively. For each pair of state s and action a, we define two Q values: the adversarial Q value denoted by $Q_\pi^{\mathcal{A}}(s,a)$ and the honest Q value denoted by $Q_\pi^{\mathcal{H}}(s,a)$, where $Q_\pi^{\mathcal{A}}(s,a)$ and $Q_\pi^{\mathcal{H}}(s,a)$ represent the expected discounted accumulated reward for the attacker and honest proposers, respectively, resulting from taking action a in state s under strategy π.

$$\begin{aligned} Q_\pi^{\mathcal{A}}(s,a) &= \mathbf{E}\big[\sum_{k=0}^{\infty} \gamma^k r_{t+k}^{\mathcal{A}} | s_t = s, a_t = a, \pi\big], \ \text{and} \\ Q_\pi^{\mathcal{H}}(s,a) &= \mathbf{E}\big[\sum_{k=0}^{\infty} \gamma^k r_{t+k}^{\mathcal{H}} | s_t = s, a_t = a, \pi\big] \ . \end{aligned} \tag{9.5}$$

DQL uses two neural networks to estimate the Q-values. The first network is the Q-network, denoted by $Q_{\texttt{estimate}}$, which is responsible for estimating the

Table 9.1: Block ratio comparison in a full-predictable protocol ($\eta = 0$)

stake share	0.3	1/3	0.35	0.4	0.45
$\pi^{\texttt{SP1}}$	0.3658	0.4285	0.4635	0.5882	0.7563
$\pi^{\texttt{SP2}}$	0.3720	0.4375	0.4740	0.6031	0.7720
$\pi^{\texttt{0-SP}}$	0.3722	0.4383	0.4753	0.6054	0.7766

Q-values. The weights of the Q-network are constantly getting updated after each iteration. The second network is the target network denoted by $Q_{\texttt{target}}$, which is responsible for calculating the target Q-values. The parameters of the target network are updated with respect to the Q-network after a predetermined number of steps. Both the Q-network and the target network take a state as their input, which indicates that the number of input nodes is equal to the dimension size of the state space. The number of output nodes in these networks is equal to the total number of available actions, with each output node representing the estimated or target Q-value for the given state under a specific action.

To calculate the block ratio of a selfish proposer in a blockchain environment, both honest and adversarial Q-values need to be estimated. Therefore, in total, we need to implement four separate neural networks: the adversarial $Q_{\texttt{estimate}}$ and $Q_{\texttt{target}}$ as well as the honest $Q_{\texttt{estimate}}$ and $Q_{\texttt{target}}$.

The agent, which is the selfish proposer, follows the ϵ-greedy algorithm to select the action at each state. As explained in [23], to select the greedy action, we define an objective function as follows:

$$O(s,a) = \frac{Q^{\mathcal{A}}(s,a)}{Q^{\mathcal{A}}(s,a) + Q^{\mathcal{H}}(s,a)} \ . \tag{9.6}$$

Here, we explain the process of training the adversarial networks. The same process is used to train the honest networks. Assume $e_t = (s_t, a_t, s_{t+1}, r_t^{\mathcal{A}}, r_t^{\mathcal{H}})$ represents a state transition experience according to which we want to train the adversarial networks. First, the estimated Q-value is calculated as $Q^{\mathcal{A}}_{\texttt{estimate}}(s_t, a_t)$. Then, the target network is used to calculate the target Q-value, which is equal to $r_t^{\mathcal{A}} + \gamma Q^{\mathcal{A}}_{\texttt{target}}(s_{t+1}, a')$. Here, a' is the action that maximizes the objective function at state s_{t+1}, i.e., $a' = \texttt{argmax}_a O(s_{t+1}, a)$. Once the estimated and the target Q-values are calculated, a loss function is used to measure the difference between these two values. The loss is then backpropagated to the adversarial Q-network to update the network parameters. For more information regarding the implementation part, readers are referred to [19].

In the following, we explain the state format for the implementation of the selfish proposing attack in different LC-PoS blockchain environments.

9.4.2 DQL Implementation of Nothing-at-Stake Selfish Proposing

In our implementation, each state in LC-PoS blockchains with perfect randomness has the following format:

$$s_t = (l_{\mathcal{A}}, l_{\mathcal{H}}, n_{\mathcal{A}}^{l_{\mathcal{H}}-k_1+1}, \cdots, n_{\mathcal{A}}^{l_{\mathcal{H}}-1}, n_{\mathcal{A}}^{l_{\mathcal{H}}}, \texttt{publish}, \texttt{match}, \texttt{latest}) \ , \tag{9.7}$$

where $l_{\mathcal{A}}$ represents the adversarial fork length, $l_{\mathcal{H}}$ represents the honest fork length, $n_{\mathcal{A}}^{i}$ represents the number of adversarial blocks proposed after the time slot at which the ith block in the honest fork is proposed, `publish` represents the number of blocks in the adversarial fork that are published to the honest network, `match` represents whether the action `match` is active or not, and `latest` represents whether the latest block is proposed by the honest proposers or not. Note that this state representation fits both single-proposer and multi-proposer models. The parameter k_1 represents the number of blocks in the honest fork for which information regarding their subsequent time slots is stored in the state.

As mentioned in Section 9.2.2, each action consists of two subactions, where the first subaction is `jump`, and the second one is chosen among 3 different possibilities: `override`, `match`, and `wait`. The number of possibilities for subaction `jump` gets limited by the value of k_1. The state stores information regarding the current fork as well as information regarding the latest k_1 honest blocks. Therefore, the agent can either continue working on top of the current fork or jump on top of one of those k_1 honest blocks and create a new fork. As a result, the total number of available actions that the agent can take is equal to $3(k_1 + 1)$. This implies that the implemented neural networks should have $3(k_1 + 1)$ outputs, where each output represents the Q-value of the input state under one of the available actions. It is worth mentioning that each state has its own set of possible actions, which is a subset of the whole available actions.

9.4.3 DQL Implementation of Full-Predictable Selfish Proposing

In our implementation, each state in full-predictable LC-PoS blockchains has the following format:

$$\begin{aligned} s_t = &(l_{\mathcal{A}}, l_{\mathcal{H}}, n_{\mathcal{A}}^{l_{\mathcal{H}}-k_1+1}, \cdots, n_{\mathcal{A}}^{l_{\mathcal{H}}-1}, n_{\mathcal{A}}^{l_{\mathcal{H}}}, \texttt{publish}, \texttt{match}, \texttt{latest}, \\ &h^1, h^2, \cdots, h^{k_2}) \ . \end{aligned} \tag{9.8}$$

In the state representation above, the first part is similar to the state representation of LC-PoS protocols with perfect randomness. The second

part contains information regarding the proposers of future slots. Due to the full predictability, we can assume each slot has exactly one proposer. k_2 denotes the number of future slots whose proposers can be predicted at the current slot. h^i represents the height of the chain in the ith upcoming time slot. Let h^0 (the height of the current slot) be defined to equal 0. In this case, h^i can be obtained as follows:

$$h^i = \begin{cases} h^{i-1} + 1 & \text{if the } i^{\text{th}} \text{ upcoming time slot is adversarial.} \\ h^{i-1} - 1 & \text{if the } i^{\text{th}} \text{ upcoming time slot is honest.} \end{cases} \tag{9.9}$$

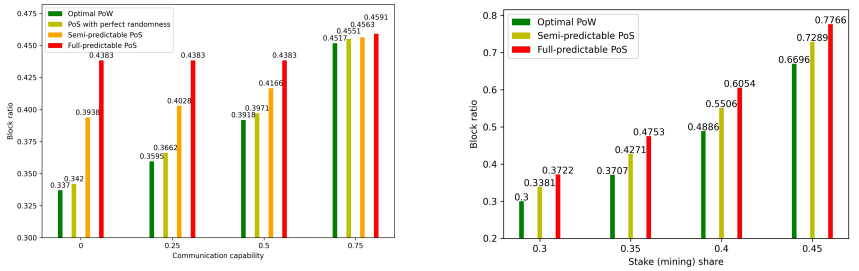
9.4.4 DQL Implementation of Semi-Predictable Selfish Proposing

In our implementation, each state in semi-predictable LC-PoS blockchains has the following format:

$$\begin{aligned} s_t = &(l_{\mathcal{A}}, l_{\mathcal{H}}, n_{\mathcal{A}}^{l_{\mathcal{H}}-k_1+1}, \cdots, n_{\mathcal{A}}^{l_{\mathcal{H}}-1}, n_{\mathcal{A}}^{l_{\mathcal{H}}}, \texttt{publish}, \texttt{match}, \texttt{latest}, \\ &\Delta_{\mathcal{A}}^1, \Delta_{\mathcal{A}}^2, \cdots, \Delta_{\mathcal{A}}^{k_3}) \ . \end{aligned} \tag{9.10}$$

In the state representation above, the first part is similar to the state representation of LC-PoS protocols with perfect randomness. The second part contains information regarding the future adversarial slots. k_3 represents the number of future adversarial blocks whose proposing time slot can be predicted at the current time slot. $\Delta_{\mathcal{A}}^i$ represents the difference between the slot number at which the ith future adversarial block will be proposed and the current slot number.

In Cardano, which can be considered the most well-known LC-PoS protocol, proposers use a verifiable random function (VRF) to determine the future slot proposers. Since the randomness used in the Cardano VRF function is extracted from the previous epoch, Cardano can be characterized as a semi-predictable LC-PoS protocol. Our DQL implementation to analyze the selfish proposing attack within various environments is presented in [1]. The graph depicted in Figure 9.4a shows a comparison of the block ratios achieved by a selfish proposer (miner) with a stake (mining) share equal to 1/3 under various environments and different communication capabilities. As can be seen in Figure 9.4a, for selfish proposers with limited communication capabilities, the predictability of future slot proposers can lead to a significant increase in the selfish proposing block ratio. The graph depicted in Figure 9.4b shows a comparison of the block ratios achieved by a selfish proposer (miner) with communication capability $\eta = 0$ under various environments and different stake (mining) shares. More details regarding implementation are presented in Appendix 9.A9.1.



(a) Stake (mining) share $\alpha = \frac{1}{3}$ (b) Communication capability $\eta = 0$

Figure 9.4: Block ratio comparison

To move towards a real-world scenario, in Appendix 9.A9.2, we present an implementation for the selfish proposing attack within an environment that encompasses the location of proposer nodes. The implementation of network node locations can help us analyze the selfish proposing attack while accounting for the complexities of real-world network conditions.

9.5 Conclusion

In this paper, we analyzed the selfish proposing attack in LC-PoS protocols, using both theoretical and implementation-based methods. The primary takeaway from this paper is that selfish proposing in LC-PoS protocols is more destructive compared to selfish mining in PoW protocols. Our analysis showed that while the nothing-at-stake phenomenon has a slight effect on a selfish proposer's block ratio, predictability can significantly increase it, especially when the attacker's communication capability is relatively low. In addition to providing theoretical analysis for selfish proposing in simple environments, such as full-predictable LC-PoS protocols and LC-PoS protocols with perfect randomness, we have introduced a deep Q-learning-based tool that enables us to analyze selfish proposing attacks in more complex environments, such as semi-predictable LC-PoS protocols. Using this tool, we can analyze the selfish proposing attack in a realistic environment, considering specifications such as node locations and the transaction pool structure.

Bibliography

[1] Implementaion of the selfish proposing attack. https://github.com/RoozbehSrnch/Selfish-Proposing-Attack.

[2] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[3] Vivek Bagaria, Amir Dembo, Sreeram Kannan, Sewoong Oh, David Tse, Pramod Viswanath, Xuechao Wang, and Ofer Zeitouni. Proof-of-stake longest chain protocols: Security vs predictability. In *Proceedings of the 2022 ACM Workshop on Developments in Consensus*, pages 29–42, 2022.

[4] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. Sok: Consensus in the age of blockchains. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 183–198, 2019.

[5] Roi Bar-Zur, Ameer Abu-Hanna, Ittay Eyal, and Aviv Tamar. Werlman: to tackle whale (transactions), go deep (rl). In *Proceedings of the 15th ACM International Conference on Systems and Storage*, pages 148–148, 2022.

[6] Roi Bar-Zur, Danielle Dori, Sharon Vardi, Ittay Eyal, and Aviv Tamar. Deep bribe: Predicting the rise of bribery in blockchain mining with deep rl. *Cryptology ePrint Archive*, 2023.

[7] Jonah Brown-Cohen, Arvind Narayanan, Alexandros Psomas, and S Matthew Weinberg. Formal barriers to longest-chain proof-of-stake protocols. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 459–473, 2019.

[8] Hongwei Chen. Interesting series associated with central binomial coefficients, catalan numbers and harmonic numbers. *J. Integer Seq.*, 19(1):16–1, 2016.

[9] Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In *Financial Cryptography and Data Security: 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*, pages 23–41. Springer, 2019.

[10] Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29-May 3, 2018 Proceedings, Part II 37*, pages 66–98. Springer, 2018.

[11] Amir Dembo, Sreeram Kannan, Ertem Nusret Tas, David Tse, Pramod Viswanath, Xuechao Wang, and Ofer Zeitouni. Everything is a race and nakamoto always wins. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 859–878, 2020.

[12] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102, 2018.

[13] Matheus VX Ferreira and S Matthew Weinberg. Proof-of-stake mining games with perfect randomness. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 433–453, 2021.

[14] Johannes Göbel, Holger Paul Keeler, Anthony E Krzesinski, and Peter G Taylor. Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay. *Performance evaluation*, 104:23–41, 2016.

[15] LM Goodman. Tezos—a self-amending crypto-ledger white paper. *URL: https://www. tezos. com/static/papers/white paper. pdf*, 4:1432–1465, 2014.

[16] Cyril Grunspan and Ricardo Pérez-Marco. On profitability of selfish mining. *arXiv preprint arXiv:1805.08281*, 2018.

[17] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual international cryptology conference*, pages 357–388. Springer, 2017.

[18] Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[20] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, 2008.

[21] Joachim Neu, Ertem Nusret Tas, and David Tse. Two more attacks on proof-of-stake ghost/ethereum. In *Proceedings of the 2022 ACM Workshop on Developments in Consensus*, pages 43–52, 2022.

[22] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *Financial Cryptography and Data Security: 20th International Conference, FC 2016, Christ Church, Barbados, February 22–26, 2016, Revised Selected Papers 20*, pages 515–532. Springer, 2017.

[23] Taotao Wang, Soung Chang Liew, and Shengli Zhang. When blockchain meets ai: Optimal mining strategy achieved by machine learning. *International Journal of Intelligent Systems*, 36(5):2183–2207, 2021.

[24] Roi Bar Zur, Ittay Eyal, and Aviv Tamar. Efficient mdp analysis for selfish-mining in blockchains. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 113–131, 2020.

Appendix

9.A1 Preliminaries

9.A1.1 Selfish Mining in PoW

In selfish mining, once a malicious miner mines a new block, he keeps the block secret and does not publish it immediately. Therefore, the other miners will continue mining on top of the chain that is no longer the longest chain. By performing selfish mining, some part of honest mining power gets wasted, and the attacker can increase his relative revenue. However, the selfish miner who keeps his block secret cannot make sure that his block will be added to the main chain in the future since the honest chain may manage to orphan the selfish miner's secret chain. Therefore, it is of huge importance that the selfish miner follows a proper strategy in each state of the chain race [12]. The authors in [22], proposed an algorithm to find the optimal strategy for selfish mining in Bitcoin, which uses the Markov Decision Process (MDP) to find the best possible action in each state based on the attacker's mining power and communication capability. The proposed solution is not Bitocin-specific and can be applied to all other PoW-based blockchains. As mentioned in this paper, each state of selfish mining in PoW protocols can be represented using a tuple $(l_a, l_h, \texttt{fork})$, where l_a denotes the length of the attacker's chain, l_h is the length of the honest chain, and `fork` gives information regarding the miner of the latest block. The set of actions is composed of four different actions, which are `adopt`, `override`, `match`, and `wait`. `adopt` means the selfish miner leaves his secret chain and continues mining on top of the honest chain. `override` represents that the attacker publishes his secret chain that is longer than the honest chain. `match` means once the honest miners mine a new block, the attacker publishes a conflicting block with the same height. And finally, `wait` means that the attacker continues mining on top of his secret chain.

9.A1.2 Deep Q-Learning

The goal of reinforcement learning, which is one of the fields of AI, is to produce an autonomous agent that interacts with the environment to learn the optimal policy, i.e., the policy that can maximize the long-term reward, through trial and error [2]. Q-learning is a reinforcement learning technique that utilizes

Markov Decision Process (MDP) to find the optimal action for each state. At each time step t, the agent observes a state s_t and takes an action a_t from the action space. As a result of this action, the agent receives a scalar reward r_t and moves to the next step s_{t+1}, according to the environment's behavior. The discounted accumulated reward at time step t is defined as $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$, where $\gamma \in (0, 1]$ is the discount factor. In Q-learning, for each pair of state s and action a, we define a quality function $Q_\pi(s, a) = \mathbf{E}\big[R_t | s_t = s, a_t = a, \pi\big]$ that represents the expected discounted accumulated reward for taking action a in state s and then following policy π. An optimal state-action quality function is the maximum expected discounted accumulated reward achievable by any policy at state s for action a. To calculate the optimal state-action quality values, the agent stores the Q-value of each state-action pair in a table and uses the following equation to update the table after each iteration with the environment:

$$Q(s_t, a_t) = (1 - \beta)Q(s_t, a_t) + \beta[r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})] \ , \tag{9.11}$$

where $\beta \in (0, 1]$ is the learning rate [18]. For more details regarding Q-learning, readers are referred to [18, 23]. One of the main limitations of Q-learning is that it is applicable only to environments dealing with a limited number of states and actions since storing the quality value for each pair of state-action is infeasible for high-dimensional settings. Deep learning enables reinforcement learning to scale to high-dimensional states and action spaces [2]. Deep Q-learning [19], introduced by Mnih et al. in 2015, uses neural networks rather than look-up tables to approximate the state-action quality values. The neural network takes a state as its input and outputs a state-action quality value for each action in the action space. To properly approximate the quality function, the neural network needs to be trained for a sufficient number of iterations. At each iteration, a loss function is used to compare the result of predicted output value $Q(s_t, a_t)$ with the target value $r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$. The neural network weights get updated with respect to the loss function. Reinforcement learning was known to face instability and divergence issues when a neural network is used to implement the Q function. To solve these issues, the authors in [19] have used two main ideas, implementing the experience replay mechanism and using two separate neural networks to implement the Q function and the target value. In the experience replay mechanism, the agent stores its interaction experiences with the environment at each time step, i.e., $e_t = (s_t, a_t, r_t, s_{t+1})$, in a database. In each learning step, a mini-batch of these experiences is randomly selected and used to train the neural network. By using the experience replay, the network can learn the new experiences without forgetting about the old ones. Moreover, to reduce the correlation between the predicted output value and the target value, two separate neural networks are trained to approximate the value and the target functions. The weights of the value function get updated

at every iteration; however, the parameters of the target function get updated with respect to the value function after a pre-determined number of steps.

9.A2 Proof of Stake

The model we use to analyze PoS protocols in this paper is built upon the model introduced in [7]. In this section, we first present a brief summary of the model introduced in [7] and then extend the model with some new definitions and notations.

9.A2.1 Overview of the Model Introduced in [7]

In each PoS protocol, there exists a set of proposers who follow the protocol to maintain and extend a blockchain ledger. A blockchain ledger consists of a chain of blocks, where each block is cryptographically connected to its previous block and stores pieces of semantic information such as transactions. Time in PoS protocols is divided into a set of slots represented by t. Each proposer, which is denoted by P, owns one or more pairs of public and private keys and has a local view. The local view of proposer P at time slot t contains a set of chains that proposer P is aware of at the start of time slot t. The basic unit of stake in PoS protocols is called a coin and is denoted by c. In the blockchain ledger, each coin belongs to an account denoted by $\texttt{acct}$. Each account, which may contain multiple coins, is linked to a public key and belongs to the proposer who owns the corresponding private key of that public key. A transaction enables the transfer of a coin from one account to another account. The proposers are responsible for proposing new blocks to extend the blockchain ledger. We use notation $\texttt{Proposer}(B)$ to represent the proposer of block B. Each block B contains three main items: a pointer to its previous (parent) block denoted by $\texttt{Parent}(B)$, a time slot t_B at which the block is created, and a reference to an account $\texttt{acct}_B$ that witnesses the block validity [7]. A block may contain other items such as content that includes the transactions.

9.A2.2 Formal Model to Define PoS Protocols

Let $\Pi^{\texttt{PoS}}$ denote a PoS protocol. We use notations $\texttt{CH}$, and $\texttt{Head}(\texttt{CH})$ to represent a chain and its rightmost (most recent) block, respectively. $B \in \texttt{CH}$ represents that block B is included in $\texttt{CH}$. $\texttt{CH}_1 \subseteq \texttt{CH}_2$ represents that chain $\texttt{CH}_1$ is a subchain of chain $\texttt{CH}_2$. If $\texttt{CH}_1$ results from pruning the k rightmost blocks of $\texttt{CH}_2$, we use the notation $\texttt{CH}_1 = \texttt{Prune}_k(\texttt{CH}_2)$.

In PoS protocols, the eligibility of proposers for proposing new blocks is assessed based on the coins they own. The probability of a proposer being selected as the block proposer is proportional to the total number of coins available in the proposer's accounts. At the start of each slot, a set[2] of accounts is randomly selected, whose owners are eligible to propose a new block. The random selection of accounts enhances the blockchain's decentralization and strengthens its resistance against censorship. Once a proposer is eligible to propose a block, the proposer needs to follow the fork-choice rule to select the best chain available in his local view and then propose the new block on top of the selected chain. The fork-choice rule is essential to guarantee the blockchain's liveness [21]. Different PoS protocols may define different fork-choice rules, leading to distinct interpretations of the term "best chain". Here, we define the longest-chain fork-choice rule.

Definition 9.A2.1 (Longest-chain)**.** Let $\texttt{chainSet}_t$ represent the set of valid chains available in the view of proposer P at time slot t. According to the longest-chain fork-choice rule, the best chain available in $\texttt{chainSet}_t$ is the chain whose length is greater compared to the other chains available in $\texttt{ChainSet}_t$.

To precisely explain how proposers are selected, the concept of coin ownership should be discussed. Since multiple proposers may have owned a specific coin c throughout the history of blockchain, it is of huge importance to determine which proposer can claim to own the coin c at time slot t. We first define the term coin owner with respect to a specific chain.

Definition 9.A2.2 (Coin owner with respect to chain CH)**.** For a given coin c and a chain CH, coin c belongs only to the last account it has visited in CH. If assuming that the last visited account is linked to a public key PubK, the owner of coin c with respect to chain CH, denoted by $\texttt{Owner}_{\texttt{CH}}(c)$, is the proposer who owns the corresponding private key of the public key PubK [7].

Additionally, we need to define the term coin owner with respect to a specific time slot, which is a different concept from Definition 9.A2.2. For a given coin c at time slot t, we define two types of ownership: a live owner and a frozen owner. The coin live owner at time slot t can be considered as the most recent owner of the coin with respect to the best available chain at time slot t. Whereas the coin frozen owner at time slot t is the owner of the coin with respect to a specific subchain of the best available chain at time slot t.

Definition 9.A2.3 (Coin live owner at slot t)**.** Let $\texttt{CH}^{\texttt{best}}$ be the best chain in the view of proposer P at time slot t. The live owner of coin c at time slot t in the view of proposer P is $\texttt{Owner}_{\texttt{CH}^{\texttt{best}}}(c)$.

[2]Based on the PoS protocol, the account set corresponding to a specific time slot can be empty or contain one or more number of accounts.

PoS protocols define a stake reference subchain, which is a pruned version of the best chain, and use this subchain to specify frozen coin owners at each time slot.

Definition 9.A2.4 (Stake reference subchain)**.** The stake reference subchain of chain $\mathtt{CH}$ at time slot t, which is denoted by $\mathtt{RCH}_t^{\mathtt{stake}}$, is a subchain of $\mathtt{CH}$ resulting from pruning its k rightmost blocks, i.e., $\mathtt{RCH}_t^{\mathtt{stake}} = \mathtt{Prune}_k\big(\mathtt{CH}\big)$. Here, k can be calculated using a deterministic function $f_{\mathtt{prune}}^{\mathtt{stake}}$ defined by protocol $\Pi^{\mathtt{PoS}}$.

The function $f_{\mathtt{prune}}^{\mathtt{stake}}$ takes as input a time slot t and a chin $\mathtt{CH}$ and outputs the number of blocks to be pruned. Function $f_{\mathtt{prune}}^{\mathtt{stake}}$ can vary in different PoS protocols.

Definition 9.A2.5 (Coin frozen owner at slot t)**.** Let $\mathtt{RCH}_t^{\mathtt{stake}}$ be the corresponding stake reference subchain of the best chain in the view of proposer P at time slot t. The frozen owner of coin c at time slot t in the view of proposer P is $\mathtt{Owner}_{\mathtt{RCH}^{\mathtt{stake}}}(c)$.

In PoS protocols, the concept of frozen coin owner is used to assess the eligibility of proposers to propose blocks. In other words, the probability of an account owner being selected as the block proposer is proportional to the account's frozen balance, where the frozen balance is calculated based on the number of coins belonging to that account with respect to the stake reference subchain.

Definition 9.A2.6 (Coin-frozen owner mapping)**.** Let $\mathtt{RCH}_t^{\mathtt{stake}}$ be the corresponding stake reference subchain of the best available chain in the view of proposer P at time slot t. The coin-frozen owner mapping in the view of proposer P at time slot t, which is denoted by M_t^P, represents the mapping between the set of coins available in the chain $\mathtt{RCH}_t^{\mathtt{stake}}$ and their owners with respect to the chain $\mathtt{RCH}_t^{\mathtt{stake}}$.

In addition to the stake reference subchain, PoS protocols define another concept as the seed reference subchain. PoS protocols rely on pseudo-random seeds extracted from the ledger to randomly select a set of proposers who are eligible to propose new blocks.

Definition 9.A2.7 (Seed reference subchain)**.** The seed reference subchain of chain $\mathtt{CH}$ at time slot t, which is denoted by $\mathtt{RCH}_t^{\mathtt{seed}}$, is a subchain of $\mathtt{CH}$ resulting from pruning its k' rightmost blocks, i.e., $\mathtt{RCH}_t^{\mathtt{seed}} = \mathtt{Prune}_{k'}\big(\mathtt{CH}\big)$. Here, k' can be calculated using a deterministic function $f_{\mathtt{prune}}^{\mathtt{seed}}$ defined by protocol $\Pi^{\mathtt{PoS}}$.

The function $f_{\mathtt{prune}}^{\mathtt{seed}}$ takes as input a time slot t and a chain $\mathtt{CH}$ and outputs the number of blocks to be pruned. The seed reference subchain is used to extract a random seed for time slot t.

Definition 9.A2.8 (Time slot seed)**.** Let $\mathtt{CH}^{\mathtt{best}}$ and $\mathtt{RCH}_t^{\mathtt{seed}}$ be the best chain and its corresponding seed reference subchain in the view of proposer P at time slot t. The seed of time slot t in the view of proposer P, which is denoted by s_t^P, can be calculated using a deterministic function $f^{\mathtt{random}}$ defined by protocol $\Pi^{\mathtt{PoS}}$, which takes the chain $\mathtt{RCH}_t^{\mathtt{seed}}$ as its input and extracts a pseudo-random number as its output.

For each time slot in PoS protocols, a set of accounts is selected, whose owners are eligible to propose a block at that time slot.

Definition 9.A2.9 (Eligible account set)**.** Let s_t^P and M_t^P be the corresponding seed and the coin-frozen owner mapping of time slot t in the view of proposer P. The eligible account set of time slot t in the view of proposer P, denoted by $\mathtt{acctSet}_t^P$, represents the set of accounts whose owners are eligible to propose a block at time slot t with respect to the view of proposer P. The eligible coin set $\mathtt{acctSet}_t^P$ can be obtained using the deterministic function $f^{\mathtt{acctSet}}$ defined by protocol $\Pi^{\mathtt{PoS}}$, which takes time slot seed s_t^P and the coin-frozen owner mapping M_t^P as its inputs and outputs a randomly selected set of accounts.

Based on how the eligible account set is defined, PoS protocols can be divided into two groups: the single-proposer model and the multi-proposer model.

Definition 9.A2.10. In a general categorization, PoS protocols are divided into one of the following models:

- **Single-proposer model**: for each time slot t, the eligible account set exactly contains a single account.
- **Multi-proposer Model**: for each time slot t, the eligible account set can be empty or contain one or more accounts.

One can consider the process of determining the eligible account set as an account selection lottery held at each slot. The randomness of the time slot seed results in a different lottery output for each slot. If in a longest-chain PoS protocol, the time slot seed is extracted from the best available chain rather than the seed reference subchain, a malicious proposer would be able to run as many account selection lotteries as the number of blocks in the block tree at each slot. This would increase the malicious proposer's chance to build a longer chain compared to the honest chain. To understand how the security of longest-chain PoS protocols is affected by the number of pruned blocks of the seed reference subchain, i.e., k', readers can refer to [3]. There exists an attack in the PoS blockchain called the coin-grinding attack that can be applied to the protocols where $\mathtt{RCH}_t^{\mathtt{seed}} \subseteq \mathtt{RCH}_t^{\mathtt{stake}}$. In this attack, since a malicious proposer can get aware of the time slot seed earlier than the coin-frozen owner mapping of that time slot, the malicious proposer can transfer coins between his accounts

to increase his winning chance in the account selection lottery. To mitigate the coin-grinding attack, the PoS protocol should be designed in a way that $\mathtt{RCH}_t^{\mathtt{stake}} \subset \mathtt{RCH}_t^{\mathtt{seed}}$.

Each PoS protocol $\Pi^{\mathtt{PoS}}$ can be defined using two deterministic functions: a validating function $\mathtt{Validate}(\cdot)$ and a proposing function $\mathtt{Propose}(\cdot)$. Function $\mathtt{Validate}(\cdot)$ verifies the validity of blocks, and function $\mathtt{Propose}(\cdot)$ is used to propose new blocks and extend the blockchain. In the following, we will define these two functions. Note that our definition of validating function is almost the same as the definition presented in [7] with the difference that we use the concept of frozen coin owner to specify proposers.

Definition 9.A2.11 (Validating function)**.** Function $\mathtt{Validate}(\cdot)$ takes a block as input and determines whether the block is valid or not. Note that each block $B \in \mathtt{CH}$ includes a pointer to its parent $\mathtt{Parent}(B)$, a time slot t_B at which the block is proposed, a reference to an account $\mathtt{acct}_B$, and a signature [3] σ_B proving the ownership over the account $\mathtt{acct}_B$. Block B is valid at time slot t from the point of view of proposer P if and only if:

- $\mathtt{Parent}(B)$ is valid.
- Assume $\mathtt{CH}^{\lceil B}$ represent a subchain of $\mathtt{CH}$ resulting from pruning its rightmost blocks up to including block B. Assume further that $\mathtt{acctSet}_{t_B}$ be the corresponding eligible account set of time slot t_B with respect to the chain $\mathtt{CH}^{\lceil B}$ [4]. Then

$$\mathtt{acct}_B \in \mathtt{acctSet}_{t_B} \tag{9.12}$$

and

$$\sigma_B \text{ is a valid signature.} \tag{9.13}$$

- $t_B \in (t_{\mathtt{Parent}(B)}, t]$

Definition 9.A2.12 (Proposing function)**.** The function $\mathtt{Propose}(\cdot)$ takes as input a chain $\mathtt{CH}$, a time slot t, an account $\mathtt{acct}$, and a private key $\mathtt{PvtK}$ corresponding to the account $\mathtt{acct}$. For any account $\mathtt{acct}$ at anytime t, if there exists a valid block B, where $\mathtt{Parent}(B) = \mathtt{Head}(\mathtt{CH})$, $t_B = t$, and $\mathtt{acct}_B = \mathtt{acct}$, then $\mathtt{Propose}(\mathtt{CH}, t, \mathtt{acct}, \mathtt{PvtK}) = B$; otherwise, $\mathtt{Propose}(\mathtt{CH}, t, \mathtt{acct}, \mathtt{PvtK}) = \perp$.

If in a PoS protocol, the seed and the coin-frozen owner mapping of a time slot are known prior to the start of the time slot, the proposers can specify the eligible account set of that time slot before its start. This phenomenon is called predictability in PoS protocols.

[3] The signature should be applied to a piece of information related to the current block.

[4] $\mathtt{CH}^{\lceil B}$ can be considered as the best available chain in the view of $\mathtt{Proposer}(B)$ at time slot t_B.

Definition 9.A2.13 (D-semi predictable)**.** A PoS protocol is called D-semi predictable if at any given time slot t, any proposer P can predict whether his own accounts belong to $\texttt{acctSet}^P_{t+D'}$ for all $0 < D' \leq D$.

Definition 9.A2.14 (D-full predictable)**.** A PoS protocol is called D-full predictable, if at any given time slot t, any proposer P can predict $\texttt{acctSet}^P_{t+D'}$ for all $0 < D' \leq D$.

To (partially) predict the eligible account set at time slot t, the seed and the coin-frozen owner mapping of time slot t should be known to the proposers. Therefore, having a D-semi/full predictable PoS protocol indicates that $\texttt{RCH}^{\texttt{seed}}_{t+D'} \subseteq \texttt{CH}^{\texttt{best}}_t$ and $\texttt{RCH}^{\texttt{stake}}_{t+D'} \subseteq \texttt{CH}^{\texttt{best}}_t$, for all time slot t and for all $0 < D' \leq D$.

9.A2.3 Supplementary Definitions

The following definitions are specific to the single-proposer model.

Definition 9.A2.15 (Proposing sequence)**.** For two time slots t_1 and t_2, where $t_1 \leq t_2$, the proposing sequence corresponding to the time-slot interval $T = [t_1, t_2]$, which is denoted by $\texttt{Seq}(T)$, is an ordered list of blocks that specifies the proposer of each block proposed during the interval $[t_1, t_2]$. The blocks in $\texttt{Seq}$ are ordered by the time slot at which they are proposed. $B^{\mathcal{H}}_i$ ($B^{\mathcal{A}}_i$) indicates that the ith block in the sequence $\texttt{Seq}$ is an honest (adversarial) block.

Note that not all the blocks of $\texttt{Seq}$ are guaranteed to be included in the main chain since some of them may get orphaned. In a full-predictable protocol, the attacker can predict the elements of the proposing sequence S in advance.

Definition 9.A2.16 (Honest (adversarial) proposing subsequence)**.** Each proposing sequence can be partitioned into two subsequences known as the honest and the adversarial proposing subsequences. The honest (adversarial) proposing subsequence corresponding to the interval $T = [t_1, t_2]$, which is denoted by $\texttt{Seq}^{\mathcal{H}}(T)$ ($\texttt{Seq}^{\mathcal{A}}(T)$), is an ordered list of honest (adversarial) blocks proposed during the interval $[t_1, t_2]$. The length of a proposing subsequence is equal to the number of blocks included in that subsequence.

For instance, let $\{B^{\mathcal{A}}_1, B^{\mathcal{A}}_2, B^{\mathcal{H}}_3, B^{\mathcal{A}}_4, B^{\mathcal{H}}_5\}$ be the proposing sequence within the time-slot interval $[1, 5]$. In this case, $\{B^{\mathcal{H}}_3, B^{\mathcal{H}}_5\}$ and $\{B^{\mathcal{A}}_1, B^{\mathcal{A}}_2, B^{\mathcal{A}}_4\}$ represent the honest and adversarial proposing subsequences within interval $[1, 5]$, respectively.

Definition 9.A2.17 (Chain race)**.** For two slots t_1 and t_2, where $t_1 \leq t_2$, the chain race corresponding to the time-slot interval $[t_1, t_2]$ is the race between the adversarial private chain and the honest public chain that satisfies the following properties:

- Before t_1, both the adversarial and honest chains share the same subchain denoted as $\mathtt{C}^{\lceil t_1}$.
- Adversarial private chain is made up of the sequence $\mathtt{C}^{\lceil t_1}||\{B_i^{\mathcal{A}}, \cdots, B_j^{\mathcal{A}}\}$, where $\{B_i^{\mathcal{A}}, \cdots, B_j^{\mathcal{A}}\}$ is the adversarial proposing subsequence corresponding to the interval $[t_1, t_2]$.
- The honest public chain is made up of the sequence $\mathtt{C}^{\lceil B_i^{\mathcal{A}}}||\{B_{i'}^{\mathrm{H}}, \cdots, B_{j'}^{\mathrm{H}}\}$, where $\{B_{i'}^{\mathrm{H}}, \cdots, B_{j'}^{\mathrm{H}}\}$ is the honest proposing subsequence corresponding to the interval $[t_1, t_2]$.

If the length of the adversarial proposing subsequence within the interval $[t_1, t_2]$ is greater than the length of its corresponding honest proposing subsequence, the attacker can **win** the chain race corresponding to the interval $[t_1, t_2]$. Winning the chain race within the interval $[t_1, t_2]$ indicates that if the attacker forks the main chain at time slot t_1 (creates a fork on top of the last block proposed before t_1), the adversarial chain within interval $[t_1, t_2]$ can orphan the honest proposers' consecutive blocks proposed within the same interval.

Definition 9.A2.18 (Longest dominant chain)**.** The longest dominant chain starting at a time slot t_1, which is denoted by $\mathtt{LDC}(t_1)$, is the chain made up of the adversarial proposing subsequence corresponding to the interval $[t_1, t_2]$, where the interval $[t_1, t_2]$ satisfies the following properties:

- $t_2 \geq t_1$.
- The attacker wins the chain race corresponding to the interval $[t_1, t_2]$.
- There is no time slot such as t_3, where $t_3 > t_2$ and the attacker wins the chain race corresponding to the interval $[t_1, t_3]$.

If there is no such a chain starting at time slot t_1, $\mathtt{LDC}(t_1)$ is called to be empty and is denoted by $\mathtt{LDC}(t_1) = \emptyset$.

We denote by $L^{\mathtt{LDC}}(t)$ the length of the longest dominant chain starting at t. If $\mathtt{LDC}(t)$ is empty, then $L^{\mathtt{LDC}}(t) = 0$. Note that if the attacker is the proposer of a block $B_t^{\mathcal{A}}$ at time slot t (slot t is adversarial), then $\mathtt{LDC}(t)$ cannot be empty, and $\min(L^{\mathtt{LDC}}(t)) = 1$. The minimum length occurs when $\mathtt{LDC}(t)$ comprises only $B_t^{\mathcal{A}}$.

9.A3 On Profitability of the Selfish Proposing Attack

In this section, we define the terms "block ratio", "relative revenue", and "time-averaged profit" and discuss the effect of selfish proposing on these terms in LC-PoS protocols. We first define the honest strategy in LC-PoS protocols.

Definition 9.A3.1 (Honest strategy)**.** A proposer always chooses to propose his new block on top of the longest chain available in his view. If the proposer manages to propose a new block, he immediately publishes the block to all the other proposers.

We assume that block proposers are divided into two groups: (1) honest proposers, denoted by $\mathcal{H}$, who follow the honest strategy, and (2) the adversarial proposers who are under the control of an attacker, denoted by $\mathcal{A}$, and may deviate from the honest strategy to gain a higher profit. The blocks that are proposed by honest proposers (the attacker) are called honest (adversarial) blocks.

Definition 9.A3.2 (Communication capability)**.** We denote by η the communication capability of attacker $\mathcal{A}$. This means, in the case of a block race, where two blocks are published simultaneously by attacker $\mathcal{A}$ and an honest proposer, the fraction of total honest proposers that receive the block proposed by the attacker first is equal to η. The honest proposers who receive the block proposed by the attacker first propose their new block on top of the attacker's block.

9.A3.1 Block Ratio

The term “bock ratio" is defined as follows:

Definition 9.A3.3 (Block ratio)**.** The block ratio of attacker $\mathcal{A}$ who follows strategy π is defined as follows:

$$\texttt{BlkRatio}_{\mathcal{A}}(\pi) = \lim_{t\to\infty} \frac{NB_t^{\mathcal{A}}}{NB_t^{\mathcal{A}} + NB_t^{\mathcal{H}}} = \lim_{t\to\infty} \frac{NB_t^{\mathcal{A}}}{NB_t^{\mathcal{T}}} \ , \tag{9.14}$$

where $NB_t^{\mathcal{A}}$, $NB_t^{\mathcal{H}}$, and $NB_t^{\mathcal{T}}$ denote the number of adversarial blocks, the number of honest blocks, and the total number of blocks in the main chain up to including time slot t, respectively.

It is obvious that selfish proposing can result in a change in the attacker's block ratio since some of the honest and adversarial blocks may get orphaned, i.e., remain out of the main chain. However, selfish proposing cannot always result in an increase in the attacker's block ratio, especially for the attackers who own a small amount of stake share. Depending on the underlying LC-PoS protocol and its detailed specifications of the longest-chain rule, the attacker needs to own a minimum threshold of stake share to make his block ratio under selfish proposing surpass that of honest proposing. To this end, we first discuss the specifications of the longest-chain rule and then define the profitable threshold.

In the longest-chain protocols, proposers always propose a new block on top of the longest chain. However, in the case when there are multiple chains with the same height, different longest-chain protocols may apply different choice rules to the same-height forks. For instance, the winning chain can be selected randomly, be the chain with a lower VRF output, or be the chain that the proposer has received first. In this paper, we assume when there exist multiple chains with the same height, the proposers choose the chain that is received in their local view earlier than the other chains.

Definition 9.A3.4 (Profitable stake (mining) share threshold)**.** For each protocol Π, the minimum amount of stake (mining) share for an attacker with communication capability η that satisfies

$$\texttt{BlkRatio}_{\mathcal{A}}(\pi^{\texttt{selfish}}) \geq \texttt{BlkRatio}_{\mathcal{A}}(\pi^{\texttt{honest}})$$

is called the profitable stake (mining) share threshold and is denoted by $\alpha^{\Pi}(\eta)$.

In this paper, we calculate the profitable stake share threshold for different LC-PoS protocols.

9.A3.2 Relative Revenue

In PoS protocols, proposers who actively participate in blockchain extension are incentivized with a reward. This reward, which is in the form of coins, can be derived from multiple sources; however, we will focus only on two of these sources: (1) the transaction fee reward, which is a number of coins paid by the transaction owner to proposers for the inclusion of his transaction in a block, and (2) the incentivizing reward, which is a number of coins paid by the protocol to proposers for generating new blocks.

Definition 9.A3.5 (Relative revenue)**.** The relative revenue of attacker $\mathcal{A}$ who follows strategy π is defined as follows:

$$\texttt{RelRevenue}_{\mathcal{A}}(\pi) = \lim_{T\to\infty} \frac{\sum_{t=1}^{T} NC_t^{\mathcal{A}}}{\sum_{t=1}^{T}\left(NC_t^{\mathcal{A}} + NC_t^{\mathcal{H}}\right)} \ , \tag{9.15}$$

where $NC_t^{\mathcal{A}}$ and $NC_t^{\mathcal{H}}$ respectively denote the number of reward coins[5] gained by the attacker and the honest proposers at time slot t[6].

[5]The coins a block proposer receives include the transaction fees as well as possibly a reward for his contribution in extending the blockchain.

[6]Note that based on the reward mechanism used in a protocol, a block proposer at slot t may receive coin(s) exactly in the same slot t or in a slot later than t. In this definition, we consider both of these scenarios as the proposer's coin(s) in slot t.

Knowing that a proposer who owns the profitable stake share threshold can increase his block ratio by applying the selfish proposing attack, the first question that needs to be addressed is whether selfish proposing can also lead to an increase in the attacker's relative revenue or not. The answer depends on the reward mechanism of the underlying LC-PoS protocol. For instance, in the first version of Ouroboros [17], i.e., the underlying protocol of Cardano, the authors have claimed that the introduced protocol is resistant to the selfish proposing attack as its reward mechanism is designed in a way that the attacker has no incentive to deviate from the protocol. In the first version of Ouroboros, the epoch reward is distributed among the proposers proportional to their stake shares rather than their block ratio in the epoch. In other words, the reward mechanism is not sensitive to whether a slot leader (proposer) has issued a block or not in its assigned time slot. Therefore, although selfish proposing can result in a change in the attacker's block ratio, it cannot affect the reward ratio, i.e., relative revenue, the attacker receives. However, this kind of reward mechanism in which the reward is distributed according to the proposers' stake share rather than their effort on extending the chain can lead to an unfair reward distribution. An offline proposer can collect a higher amount of reward compared to an active proposer just because of owning a greater stake share. To bring fairness back to the implemented version of Cardano, the epoch reward assigned to a stake pool (proposer), which is calculated proportional to its stake share, is adjusted by the pool performance. The pool performance is defined as the block ratio divided by the stake ratio. Therefore, in the implemented version of Cardano, the relative revenue of a proposer is positively correlated with his block ratio, and selfish proposing can still threaten blockchain progress. As can be seen in the longest-chain blockchain protocols, such as Bitcoin, Cardano, etc., it seems that to fairly incentivize the proposers to participate in the blockchain extension, the reward mechanism should be designed in a way that the relative revenue of a proposer is positively correlated with his block ratio. And whenever there is a positive correlation between the relative revenue and the block ratio, the selfish proposing attack can result in an increase in the attacker's relative revenue.

9.A3.2.1 Long-Range and Short-Range Selfish Proposing

In PoS protocols, since the essence of both block reward and stake is the same thing, namely coin, the reward that a proposer collects can get added to his stake. When all the proposers follow the honest strategy, the relative revenue that a proposer collects is equal to his stake ratio. Therefore, if all the proposers add the collected reward to their stake, the proposers' stake ratios remain unchanged. However, if a malicious proposer starts the selfish proposing attack, his relative revenue becomes greater than his stake ratio. Conversely, honest

proposers' relative revenue falls below their stake share. Therefore, after adding the collected reward to their stake, the selfish proposer's stake ratio will increase, while the stake ratio of others will decrease. This implies that by applying a long-range selfish proposing attack in LC-PoS protocols, the stake share of the selfish proposer will gradually grow and eventually exceed more than one-half. Therefore, we can conclude that in an LC-PoS protocol $\Pi^{\texttt{LC-PoS}}$, a selfish proposer $\mathcal{A}$ who owns stake share $\alpha_{\mathcal{A}} < 51\%$ and communication capability η, can perform 51% attack at a point in the future provided that: i) proposers are rewarded proportional to the number of blocks they have contributed to the main chain, ii) protocol $\Pi^{\texttt{LC-PoS}}$ runs infinitely, and iii) $\alpha_{\mathcal{A}} > \alpha_{\eta}^{\Pi^{\texttt{LC-PoS}}}$.

To assess the selfish proposer's relative revenue in a short period of time, we define the term "short-range relative revenue".

Definition 9.A3.6 (Short-range relative revenue)**.** Let $B_1^{\texttt{Ref}}$ and $B_2^{\texttt{Ref}}$ be two consecutive stake reference blocks in the main chain. Assume attacker $\mathcal{A}$ follows strategy π during the interval $T = \big(t(B_1^{\texttt{Ref}}), t(B_2^{\texttt{Ref}})\big]$. The short-term relative revenue of attacker $\mathcal{A}$ in the interval T is defined as follows:

$$\texttt{RelRevenue}_{\mathcal{A}}\big(\pi, T\big) = \frac{\sum_{t\in T} NC_t^{\mathcal{A}}}{\sum_{t\in T}\Big(NC_t^{\mathcal{A}} + NC_t^{\mathcal{H}}\Big)} \ . \tag{9.16}$$

In this paper, we will show that with the same amount of (stake or mining) share and communication capability, selfish proposing in LC-PoS protocols can result in a higher short-term relative revenue compared to selfish mining in PoW protocols. If assuming all the blocks receive the same amount of transaction fees and the incentivizing reward is distributed exactly proportional to the proposers' block ratio, a proposer's relative revenue will be equal to his block ratio.

9.A3.3 Time-Averaged Profit

We can assess the profitability of selfish proposing using both concepts of block ratio and time-averaged profit. We first define the time-averaged profit. Note that in the following definition, we ignore the cost of block production as it is negligible in PoS protocols compared to PoW.

Definition 9.A3.7 (Time-averaged profit)**.** The time-averaged profit (profit per time slot) of attacker $\mathcal{A}$ who follows strategy π is defined as follows:

$$\texttt{Profit}_{\mathcal{A}}(\pi) = \lim_{T\to\infty} \frac{\sum_{t=1}^{T} NC_t^{\mathcal{A}}}{T} \ , \tag{9.17}$$

where $NC_t^{\mathcal{S}}$ is the number of coins the attacker received at time slot t.

Most of the papers related to PoW protocols have used the concept of relative revenue to analyze selfish mining profitability. However, the authors in papers such as [16] have argued that time-averaged profit is a better benchmark compared to the relative revenue to assess profitability. Here, we first briefly explain why the relative revenue does not reflect the whole story behind the profitability of selfish proposing in LC-PoS protocols.

Let $B_1^{\texttt{Ref}}$ and $B_2^{\texttt{Ref}}$ be two consecutive stake reference blocks in the main chain and the time slot interval T is defined as $T = \big(t(B_1^{\texttt{Ref}}), t(B_2^{\texttt{Ref}})\big]$. Assume attacker $\mathcal{A}$ has followed the honest strategy and owned stake share $\alpha > \alpha_\eta^{\Pi^{\texttt{LC-PoS}}}$ before and including the reference block $B_1^{\texttt{Ref}}$ and decides to follow selfish strategy $\pi^{\texttt{selfish}}$ in the interval T. Due to the selfish proposing attack, there is an increase in the expected value of relative revenue during the interval T, i.e.,

$$\mathbf{E}\Big[\texttt{RelRevenue}_{\mathcal{A}}^{\texttt{short}}\big(\pi^{\texttt{selfish}}, T\big)\Big] > \alpha \ . \tag{9.18}$$

However, since the amount of $\mathcal{A}$'s stake share is the same before and during the interval T, the attacker's chance of winning the proposer selection lottery is the same before and after the start of the attack. This implies that the expected number of blocks added to the main chain by attacker $\mathcal{A}$ in the interval T is equal to or even less[7] than the expected number of blocks attacker $\mathcal{A}$ contributed to the main chain during the same period of time slots before starting the attack. Therefore, an increase in the relative revenue after starting the selfish proposing attack does not necessarily reflects an increase in the selfish proposer's time-averaged profit. Using time-averaged profit, we can assess the amount of reward a selfish proposer receives regardless of the honest proposers' collected reward.

Difference in time-averaged profit between LC-PoS and PoW: In PoW protocols, there exists a difficulty adjustment mechanism (DAM) that controls transaction throughput by modifying the mining difficulty (mining target). It is shown that before a DAM, the time-averaged profit of selfish mining cannot surpass the time-averaged profit of honest mining in PoW protocols. However, once the difficulty level of mining gets adjusted, the mining puzzle gets easier and the selfish miner can mine a new block in a shorter period of time. Therefore, by applying DAM, the selfish miner's time-averaged profit increases. This shows that a selfish miner in PoW protocols should maintain the attack for a couple of difficulty periods to increase his time-averaged profit and compensate for his loss at the start of the attack.

In LC-PoS protocols, the stake share of a selfish proposer will gradually grow during the selfish proposing attack, which can guarantee an increase in the long-term time-averaged profit of the attacker. However, to assess the short-term

[7] Note that some of the adversarial blocks may get orphaned after starting the selfish proposing attack.

time-averaged profit of selfish proposing in LC-PoS protocols, we should take into account the rewarding mechanism used in the LC-PoS protocol.

Transaction reward per slot: If assuming that the total amount of transaction fee a block can receive is the same before and after the attack, the attacker's transaction reward per slot will not increase after the attack because the attacker's stake share, and consequently, his block generation rate have not changed. However, in a real-world scenario, the attacker can utilize selfish proposing to increase his transaction reward per slot. As a result of a selfish proposing attack, the block generation rate decreases, resulting in a reduction in transaction throughput. In such a competitive situation, a transaction owner who wants his transaction to get included in a block should increase the transaction fee. Therefore, there will be a rise in the amount of transaction fees, and accordingly, the attacker's transaction reward per slot will increase. Moreover, when a selfish proposer manages to orphan an honest block, the selfish proposer can steal the valuable or MEV transactions included in the honest block and put them in his own block to further enhance his time-averaged profit.

Incentivizing reward per slot: We consider two scenarios for the distribution of incentivizing rewards. In the first scenario, a block proposer who generates a new block receives an incentivizing reward through a transaction included in the block. In this case, the selfish proposing attack cannot increase the attacker's incentivizing reward per slot. In the second scenario, at the end of each `epoch`, which is a predetermined set of time slots, the incentivizing reward is distributed among proposers proportional to the number of blocks they have contributed to the main chain during the `epoch`. In this scenario, if the epoch incentivizing reward is fixed, a selfish proposer can increase his incentivizing reward per slot immediately after the start of the attack.

Trade off between transaction throughput and selfish proposer's time-averaged profit: Once an attacker starts the selfish proposing attack in an LC-PoS blockchain, due to orphan occurrence, the average number of blocks per slot will decrease, which results in a decrease in the transaction throughput. As already mentioned, in PoW protocols, there is a difficulty adjustment mechanism that aims to balance the transaction throughput by modifying the mining difficulty. Although in PoS protocols, there is no DAM, there exist some predetermined parameters such as slot duration and the average number of blocks per slot that affect the transaction throughput. In most of the PoS protocols, these parameters are determined prior to the start of the protocol; however, blockchain proposers or designers can decide to modify these parameters to balance the transaction throughput in the occurrence of selfish proposing. The transaction throughput can get balanced at the cost of an

Table 9.2: States and their corresponding sequences

State	Proposing sequence
$S_{0,0}$	-
$S_{1,0}$	$\{B_1^{\mathcal{A}}\}$
$S_{1,1}$	$\{B_1^{\mathcal{A}}, B_2^{\mathcal{H}}\}$
$S_{2,0}$	$\{B_1^{\mathcal{A}}, B_2^{\mathcal{A}}\}$
$S_{1,2}$	$\{B_1^{\mathcal{A}}, B_2^{\mathcal{H}}, B_3^{\mathcal{H}}\}$

increase in the average number of blocks per slot, which results in an increase in the selfish proposer's time-averaged profit.

9.A4 Selfish Proposing Attack in LC-PoS Protocols with Perfect Randomness

9.A4.1 Strategy $\pi^{\text{S-PR}}$

Strategy $\pi^{\text{S-PR}}$ is suitable for LC-PoS protocols with perfect randomness. We assume that the LC-PoS environment is compatible with the single-proposer model. Let α and η denote the attacker's stake share and communication capability, respectively. Let B_t^P denote the block proposed by proposer $P \in \{\mathcal{A}, \mathcal{H}\}$ at time slot t. For the sake of simplicity, the time slot at which the latest common block between the honest and adversarial chains is proposed is normalized to be 0. This normalization implies that a fork always starts at time slot $t = 1$. In our model, each state corresponds to a specific proposing sequence, where the concept of proposing sequence is introduced in Definition 9.A2.15. By following the strategy $\pi^{\text{S-PR}}$, the system transitions among five main states, each of them represented by $S_{l^{\mathcal{A}},l^{\mathcal{H}}}$, where $l_{\mathcal{A}}$ ($l_{\mathcal{H}}$) denotes the length of the adversarial (honest) fork. The corresponding proposing sequences of these 5 states are represented in Table 9.2. The selfish proposing strategy $\pi^{\text{S-PR}}$ is presented in Table 9.3. $\texttt{action}_0$, $\texttt{action}_1$, and $\texttt{action}_2$, which are mentioned in Table 9.3, are defined as follows:

$\texttt{action}_0$: The action corresponds to state $S_{0,0}$, where $l_{\mathcal{H}} = l_{\mathcal{A}} = 0$. The action represents that the attacker waits until a new block is proposed. If the new block is an honest block, the attacker adopts the honest chain and remains in the same state. If the new block is an adversarial block, the attacker moves to state $S_{1,0}$.

$\texttt{action}_1$: The action corresponds to state $S_{2,0}$, where $l_{\mathcal{H}} = l_{\mathcal{A}} - 2$. The action represents that the attacker waits until the point where his lead reduces to 1,

i.e., $l_{\mathcal{H}} = l_{\mathcal{A}} - 1$, and then publishes the whole adversarial fork to override the honest fork.

$\texttt{action}_2$: The action corresponds to state $S_{1,2}$, the sequence of which is $\{B_1^{\mathcal{A}}, B_2^{\mathcal{H}}, B_3^{\mathcal{H}}\}$. Let L denote the number of consecutive honest blocks proposed after block $B_3^{\mathcal{H}}$ and before the next adversarial block, where $L \geq 0$. This implies that the first adversarial block after visiting state $S_{1,2}$ is proposed at time slot $t = 4 + L$ and can be represented by $B_{4+L}^{\mathcal{A}}$.

1. In the case that the block proposed at time slot $t = 5 + L$ is an honest block denoted by $B_{5+L}^{\mathcal{H}}$, the attacker gives up on block $B_1^{\mathcal{A}}$ and generates a new fork (jump) on top of honest block $B_{3+L}^{\mathcal{H}}$. In this case, the new state is $S_{1,1}$, where the honest fork only contains block $B_{5+L}^{\mathcal{H}}$, and the adversarial fork only contains block $B_{4+L}^{\mathcal{A}}$.
2. In the case that the block proposed at time slot $t = 5 + L$ is an adversarial block denoted by $B_{5+L}^{\mathcal{A}}$, the attacker waits until one of the following cases happens:
   (a) If at a future time slot, the condition $l_{\mathcal{A}} = l_{\mathcal{H}} + 1$ holds, the attacker immediately publishes the whole adversarial fork to override the honest fork. In this case, the new state is $S_{0,0}$.
   (b) If at a future time slot, the condition $l_{\mathcal{A}} = l_{\mathcal{H}} - L$ holds, the attacker gives up on block $B_1^{\mathcal{A}}$ and generates a new fork (jump) on top of honest block $B_{3+L}^{\mathcal{H}}$, where the new adversarial (honest) fork contains all the adversarial (honest) blocks proposed after time slot $t = 3 + L$. By doing so, the new adversarial fork has a one-block lead over the new honest fork. Then, the attacker immediately publishes the new

Table 9.3: Strategy $\pi^{\text{S-PR}}$

current state	action	next state	probability	reward
$S_{0,0}$	$\texttt{action}_0$	$S_{0,0}$	$1-\alpha$	$R_{\mathcal{A}} = 0,\ R_{\mathcal{H}} = 1$
		$S_{1,0}$	α	$R_{\mathcal{A}} = 0,\ R_{\mathcal{H}} = 0$
$S_{1,0}$	$\texttt{wait}$	$S_{1,1}$	$1-\alpha$	$R_{\mathcal{A}} = 0,\ R_{\mathcal{H}} = 0$
		$S_{2,0}$	α	$R_{\mathcal{A}} = 0,\ R_{\mathcal{H}} = 0$
$S_{1,1}$	$\texttt{match}$	$S_{1,2}$	$(1-\eta)(1-\alpha)$	$R_{\mathcal{A}} = 0,\ R_{\mathcal{H}} = 0$
		$S_{0,0}$	$\eta(1-\alpha)$	$R_{\mathcal{A}} = 1,\ R_{\mathcal{H}} = 1$
		$S_{0,0}$	α	$R_{\mathcal{A}} = 2,\ R_{\mathcal{H}} = 0$
$S_{2,0}$	$\texttt{action}_1$	$S_{0,0}$	1	$R_{\mathcal{A}} = R_1,\ R_{\mathcal{H}} = 0$
$S_{1,2}$	$\texttt{action}_2$	$S_{1,1}$	$1-\alpha$	$R_{\mathcal{A}} = 0,\ R_{\mathcal{H}} = R_2$
		$S_{0,0}$	α	$R_{\mathcal{A}} = R_3,\ R_{\mathcal{H}} = R_4$

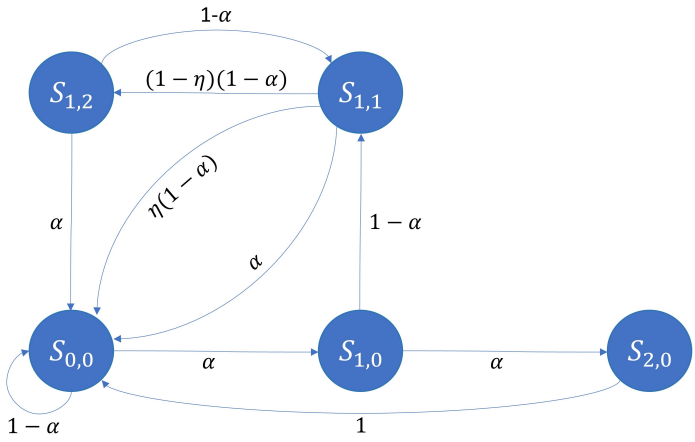


Figure 9.5: State representation of strategy $\pi^{\text{S-PR}}$

adversarial fork to override the new honest fork. In this case, the final state is $S_{0,0}$.

Note that states $S_{2,0}$ and $S_{1,2}$ contain transitions to hidden states that are not included in the five main states. The set of states and possible transitions are depicted in Figure 9.5.

9.A4.2 Block Ratio of Strategy $\pi^{\text{S-PR}}$

To obtain the block ratio achieved by following strategy $\pi^{\text{S-PR}}$, rewards R_1, R_2, R_3, and R_4, which are mentioned in Table 9.3, need to b calculated. To calculate the rewards, we use random walking on a (x, y)-grid. Throughout this paper, a two-dimensional (x, y)-grid is used to represent the chain race between the honest and adversarial forks. Each proposing sequence can be represented by a path on this grid, which is referred to as the "block path". Whenever the time slot is honest, i.e., the block proposer of the time slot is honest, the block path moves one step up. Whenever the time slot is adversarial, i.e., the block proposer of the time slot is the attacker, the block path moves one step to the right.

According to Lemma D.2. presented in [13], the adversarial reward of following $\texttt{action}_1$ at state $S_{2,0}$, i.e., R_1, can be obtained as follows:

$$R_1 = 2 + \frac{\alpha}{1-2\alpha} \quad . \tag{9.19}$$

Lemma 9.1. The average honest reward resulting from the transition from state $S_{1,2}$ to state $S_{1,1}$ under $\texttt{action}_2$ (Scenario 1 occurrence) can be obtained

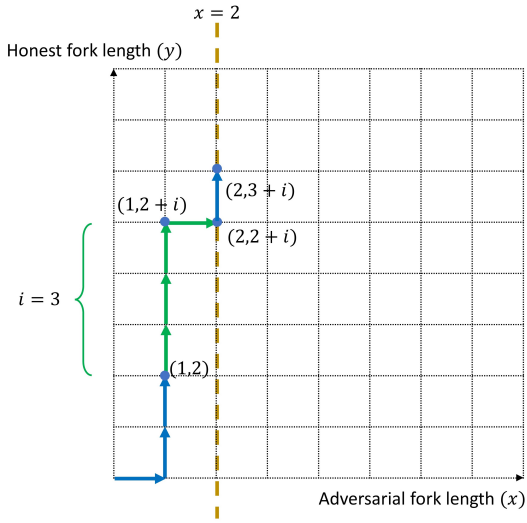


Figure 9.6: Block path in Scenario 1

as follows:

$$R_{\mathcal{H}} = R_2 = \frac{\sum_{i=0}^{\infty}(2+i)\alpha(1-\alpha)^{i+1}}{\texttt{Pr}(\text{Scenario 1})} = 2 + \frac{1-\alpha}{\alpha} \ . \tag{9.20}$$

Proof. We use a two-dimensional (x, y)-grid to represent the chain race as depicted in Figure 9.6. Being at state $S_{1,2}$ implies that the block path started at point $(0, 0)$ has reached the point $(1, 2)$. Scenario 1 is equivalent to the event that once the block path starting at point $(1, 2)$ reaches the line $x = 2$ at one of the points $(2, 2 + i)$ for $i \geq 0$, it immediately moves one step upward to reach the point $(2, 3 + i)$. The probability that the block path starting at $(1, 2)$ reaches the line $x = 2$ for the first time at point $(2, 2+i)$ and then moves to the point $(2, 3 + i)$ is equal to $\alpha(1 - \alpha)^{i+1}$. Therefore, the probability of Scenario 1 occurrence can be obtained as follows:

$$\texttt{Pr}(\text{Scenario 1}) = \sum_{i=0}^{\infty} \alpha(1-\alpha)^{i+1} = 1 - \alpha \ . \tag{9.21}$$

The adversarial reward of Scenario 1 is equal to 0. If assuming L denotes the number of consecutive honest blocks proposed after being at state $S_{1,2}$ and before the next adversarial block is proposed, the honest reward resulting from Scenario 1 is equal to $2 + L$. Therefore, the average honest reward under Scenario 1 can be obtained as follows:

$$R_2 = \frac{\sum_{i=0}^{\infty}(2+i)\alpha(1-\alpha)^{i+1}}{\texttt{Pr}(\text{Scenario 1})} = 2 + \frac{1-\alpha}{\alpha} \ . \tag{9.22}$$

□

To find R_3 and R_4, we first present a couple of lemmas.

Lemma 9.2. Let P_r denote the probability of the event that the block path starting at point $(0,0)$ reaches the line $y = x - r$ for $r \geq 1$ at least once. We have $P_r = \left(\frac{\alpha}{1-\alpha}\right)^r$.

Proof. We prove the lemma using induction. First, we find the probability that the block path reaches the line $y = x - 1$ at least once, i.e., P_1. Assume the block path reaches the line $y = x - 1$ for the first time at point $(s+1, s)$. To achieve this, the block path should reach point (s, s) without passing below the line $y = x$ and then move one step to the right. The number of paths from point $(0,0)$ to point (s, s) without passing below the line $y = x$ is equal to the sth Catalan number denoted by c_s. Therefore, the probability of reaching the line $y = x - 1$ for the first time at point $(s+1, s)$ is equal to $c_s \alpha^{s+1}(1-\alpha)^s$. Thus, the probability of reaching the line $y = x - 1$ can be calculated as follows:

$$P_1 = \alpha \sum_{s=0}^{\infty} c_s \big(\alpha(1-\alpha)\big)^s = \frac{\alpha}{1-\alpha} \ . \tag{9.23}$$

The formula for solving the series above is presented in [8]. Next, we find the probability that the block path reaches the line $y = x - 2$ at least once, i.e., P_2. Assume the block path reaches the line $y = x - 2$ for the first time at point $(s+2, s)$. To achieve this, the block path should reach point $(s+1, s)$ without passing below the line $y = x - 1$ and then move one step to the right. The number of paths from point $(0,0)$ to point $(s+1, s)$ without passing below the line $y = x - 1$ is equal to c_{s+1}. Therefore, the probability of reaching the line $y = x - 2$ for the first time at point $(s+2, s)$ is equal to $c_{s+1}\alpha^{s+2}(1-\alpha)^s$. Thus, the probability of reaching the line $y = x - 2$ can be calculated as follows:

$$P_2 = \alpha^2 \sum_{s=0}^{\infty} c_{s+1} \big(\alpha(1-\alpha)\big)^s = \frac{\alpha}{1-\alpha} \sum_{s=1}^{\infty} c_s \big(\alpha(1-\alpha)\big)^s = \left(\frac{\alpha}{1-\alpha}\right)^2 \ . \tag{9.24}$$

As the next step, assume $P_{k-1} = \left(\frac{\alpha}{1-\alpha}\right)^{k-1}$ and $P_k = \left(\frac{\alpha}{1-\alpha}\right)^k$ for $k \geq 2$. We need to prove $P_{k+1} = \left(\frac{\alpha}{1-\alpha}\right)^{k+1}$. Considering the grid depicted in Figure 9.7, we obtain:

$$P_k = \alpha P_{k-1} + (1-\alpha) P_{k+1} \implies P_{k+1} = \frac{P_k - \alpha P_{k-1}}{1-\alpha} = \left(\frac{\alpha}{1-\alpha}\right)^{k+1} \ . \tag{9.25}$$

□

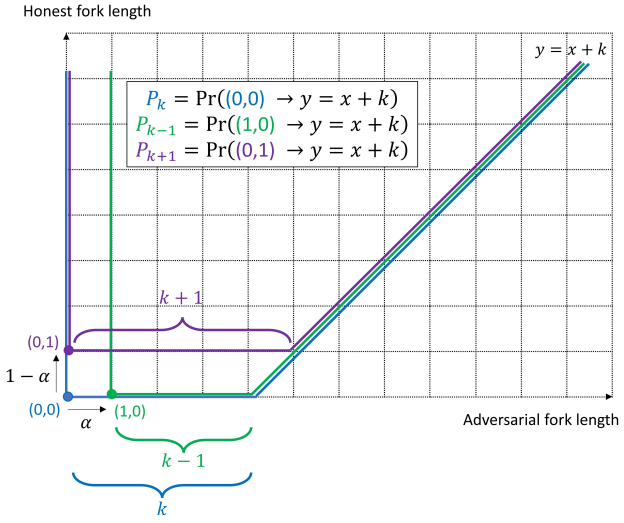


Figure 9.7: Representation of probability P_k

Lemma 9.3. Let $\overline{P_r}$ denote the probability of the event that the block path starting at $(0,0)$ never reaches the line $y = x - r$ for $r \geq 1$. We have $\overline{P_r} = 1 - \left(\frac{\alpha}{1-\alpha}\right)^r$.

Proof. The probability of the complementary event of never reaching the line $y = x - r$, i.e., P_r, is calculated in Lemma 9.2. Therefore, we have $\overline{P_r} = 1 - P_r$. □

Lemma 9.4. Let G_s^r for $r \geq 1$ denote the number of paths in a (x, y)-grid from start point $(0,0)$ to the point (s,s) without reaching the lines $y = x + 1$ and $y = x - r$ (define G_0^r to be equal to 1). We have:

$$\sum_{s=0}^{\infty} G_s^r \big(\alpha(1-\alpha)\big)^s = \Big(\frac{1}{1-\alpha}\Big) \frac{1 - (\frac{\alpha}{1-\alpha})^r}{1 - (\frac{\alpha}{1-\alpha})^{r+1}} \quad ,$$

$$\sum_{s=0}^{\infty} s G_s^r \big(\alpha(1-\alpha)\big)^s = \frac{\alpha\big(1 - (\frac{\alpha}{1-\alpha})^r\big) - r(\frac{\alpha}{1-\alpha})^r}{(1-\alpha)(1-2\alpha)\big(1 - (\frac{\alpha}{1-\alpha})^{r+1}\big)} \tag{9.26}$$

$$+ \frac{(r+1)\alpha\big(1 - (\frac{\alpha}{1-\alpha})^r\big)(\frac{\alpha}{1-\alpha})^r}{(1-\alpha)^2(1-2\alpha)\big(1 - (\frac{\alpha}{1-\alpha})^{r+1}\big)^2} \quad .$$

Proof. Assume $\texttt{Event}_1(s)$ is defined as follows: (i) the block path starting at $(0,0)$ reaches the point (s,s), and (ii) before reaching the point (s,s), the block

path never passes the line $y = x$ and never reaches the line $y = x - r$. According to the definition of G_s^r, the probability of $\texttt{Event}_1(s)$ is equal to $G_s^r \alpha^s (1-\alpha)^s$. Assume $\texttt{Event}_2(s)$ is defined as follows: (i) the block path starting at $(0,0)$ passes the line $y = x$ for the first time at point (s,s), and (ii) before reaching the point (s,s), the block path never reaches the line $y = x - r$. $\texttt{Event}_2(s)$ happens if, after the occurrence of $\texttt{Event}_1(s)$ and reaching the point (s,s), the block path immediately moves one step up to pass the line $y = x$ and reach the point $(s, s+1)$. Therefore, the probability of $\texttt{Event}_2(s)$ is equal to $G_s^r \alpha^s (1-\alpha)^{s+1}$. Assume $\texttt{Event}_3(s)$ is defined as follows: (i) the block path starting at $(0,0)$ passes the line $y = x$ for the first time at point (s,s), and (ii) the block path never reaches the line $y = x - r$ both before and after reaching the point (s,s). $\texttt{Event}_3(s)$ happens if, after the occurrence of $\texttt{Event}_2(s)$ and reaching the point $(s, s+1)$, the block path never reaches the line $y = x - r$. The event that the block path starting at $(s, s+1)$ never reaches the line $y = x - r$ is equivalent to the event that the block path starting at $(0,0)$ never reaches the line $y = x - (r+1)$, the probability of which is equal to $1 - (\frac{\alpha}{1-\alpha})^{r+1}$ according to Lemma 9.3. Therefore, the probability of $\texttt{Event}_3(s)$ is equal to $G_s^r \alpha^s (1-\alpha)^{s+1} \big(1 - (\frac{\alpha}{1-\alpha})^{r+1}\big)$. Note that since $1 - \alpha > \alpha$, a block path will eventually pass the line $y = x$. Therefore, the sum of $\texttt{Event}_3(s)$ probabilities over all values of s is equal to the probability that the block path never reaches the line $y = x - r$, the probability of which is equal to $1 - (\frac{\alpha}{1-\alpha})^r$ according to Lemma 9.3. As a result,

$$\begin{aligned} &\sum_{s=0}^{\infty} G_s^r \alpha^s (1-\alpha)^{s+1} \big(1 - (\frac{\alpha}{1-\alpha})^{r+1}\big) = 1 - (\frac{\alpha}{1-\alpha})^r \\ &\Rightarrow \sum_{s=0}^{\infty} G_s^r \big(\alpha(1-\alpha)\big)^s = \Big(\frac{1}{1-\alpha}\Big) \frac{1 - (\frac{\alpha}{1-\alpha})^r}{1 - (\frac{\alpha}{1-\alpha})^{r+1}} \ . \end{aligned} \tag{9.27}$$

To prove the second equality in Lemma 9.4, we use the variable substitution $\alpha(1-\alpha) = x$ in the equality above. We have:

$$\sum_{s=0}^{\infty} G_s^r x^s = \Big(\frac{2}{1+\sqrt{1-4x}}\Big) \frac{1 - (\frac{1-\sqrt{1-4x}}{1+\sqrt{1-4x}})^r}{1 - (\frac{1-\sqrt{1-4x}}{1+\sqrt{1-4x}})^{r+1}} \ . \tag{9.28}$$

By taking the derivative from both sides, then multiplying both sides to x, and finally substituting $x = \alpha(1-\alpha)$, the second equality in Lemma 9.13 can be obtained. □

Lemma 9.5. The probability that the block path starting at $(0,0)$ reaches the line $y = x + 1$ before reaching the line $y = x - r$ for $r \geq 1$ is equal to $\frac{1 - (\frac{\alpha}{1-\alpha})^r}{1 - (\frac{\alpha}{1-\alpha})^{r+1}}$.

Proof. The event that the block path reaches the line $y = x + 1$ for the first time at point $(s, s + 1)$ without having reached the line $y = x - r$ is the same as $\texttt{Event}_2(s)$ introduced in proof of Lemma 9.4, the probability of which is equal to $G_s^r \alpha^r (1-\alpha)^{r+1}$. The probability that the block path starting at $(0, 0)$ reaches the line $y = x + 1$ before reaching the line $y = x - r$ is equal to the sum of the $\texttt{Event}_2(s)$ probabilities over all values of s, which can be obtained as follows:

$$\sum_{s=0}^{\infty} G_s^r \alpha^s (1-\alpha)^{s+1} = \frac{1 - (\frac{\alpha}{1-\alpha})^r}{1 - (\frac{\alpha}{1-\alpha})^{r+1}} \quad . \tag{9.29}$$

□

Lemma 9.6. The probability that the block path starting at $(0, 0)$ reaches the line $y = x - r$ for $r \geq 1$ before reaching the line $y = x + 1$ is equal to $1 - \frac{1-(\frac{\alpha}{1-\alpha})^r}{1-(\frac{\alpha}{1-\alpha})^{r+1}}$.

Proof. The event that the block path starting at $(0, 0)$ reaches the line $y = x - r$ for $r \geq 1$ before reaching the line $y = x + 1$ is the complement of the event introduced in Lemma 9.5. □

Lemma 9.7. Let F_s^r for $r \geq 1$ denote the number of paths in a (x, y)-grid from start point $(0, 0)$ to the point $(s + r - 1, s)$ without reaching the lines $y = x + 1$ and $y = x - r$ (define F_0^1 to be equal to 1). We have:

$$\sum_{s=0}^{\infty} F_s^r \big(\alpha(1-\alpha)\big)^s = \Big(\frac{1}{\alpha^r}\Big)\Big(1 - \frac{1 - (\frac{\alpha}{1-\alpha})^r}{1 - (\frac{\alpha}{1-\alpha})^{r+1}}\Big) \quad ,$$

$$\sum_{s=0}^{\infty} s F_s^r \big(\alpha(1-\alpha)\big)^s = \frac{\alpha(r+1)(1 + (\frac{\alpha}{1-\alpha})^r)}{(1-\alpha)^{r+1}(1 - (\frac{\alpha}{1-\alpha})^{r+1})^2} - \tag{9.30}$$

$$\frac{2\alpha}{(1-2\alpha)(1-\alpha)^r(1 - (\frac{\alpha}{1-\alpha})^{r+1})} \quad .$$

Proof. Assume $\texttt{Event}_1(s)$ is defined as follows: (i) the block path starting at $(0, 0)$ reaches the point $(s+r-1, s)$, and (ii) before reaching the point $(s+r-1, s)$, the block path never reaches the lines $y = x + 1$ and $y = x - r$. According to the definition of F_s^r, the probability of $\texttt{Event}_1(s)$ is equal to $F_s^r \alpha^{s+r-1}(1-\alpha)^s$. Assume $\texttt{Event}_2(s)$ is defined as follows: (i) the block path starting at $(0, 0)$ reaches the line $y = x - r$ for the first time at point $(s + r, s)$, and (ii) before reaching the point $(s + r, s)$, the block path never reaches the line $y = x + 1$. $\texttt{Event}_2(s)$ happens if, after the occurrence of $\texttt{Event}_1(s)$ and reaching the point $(s + r - 1, s)$, the block path immediately moves one step to the right to pass to reach the point $(s + r, s)$, which is located in the line $y = x - r$. Therefore,

the probability of $\texttt{Event}_2(s)$ is equal to $F_s^r \alpha^{s+r}(1-\alpha)^s$. The sum of $\texttt{Event}_2(s)$ probabilities over all values of s is equal to the probability that the block path starting at $(0,0)$ reaches the line $y = x - r$ before reaching the line $y = x + 1$, the probability of which is calculated in Lemma 9.6. Therefore, we have:

$$\sum_{s=0}^{\infty} F_s^r \alpha^{s+r}(1-\alpha)^s = 1 - \frac{1 - (\frac{\alpha}{1-\alpha})^r}{1 - (\frac{\alpha}{1-\alpha})^{r+1}} \ , \tag{9.31}$$

which results in the first equality in Lemma 9.7. To prove the second equality in Lemma 9.7, we use the variable substitution $\alpha(1-\alpha) = x$ in the equality above. We have:

$$\sum_{s=0}^{\infty} F_s^r x^s = \Big(\frac{2}{1-\sqrt{1-4x}}\Big)^r \Big(1 - \frac{1 - (\frac{1-\sqrt{1-4x}}{1+\sqrt{1-4x}})^r}{1 - (\frac{1-\sqrt{1-4x}}{1+\sqrt{1-4x}})^{r+1}}\Big) \ . \tag{9.32}$$

By taking the derivative from both sides, then multiplying both sides to x, and finally substituting $x = \alpha(1-\alpha)$, the second equality in Lemma 9.7 can be obtained. □

Lemma 9.8. The average adversarial and honest rewards resulting from the transition from state $S_{1,2}$ to state $S_{0,0}$ under $\texttt{action}_2$ (Scenario 2 occurrence) can be obtained as follows:

$$\begin{aligned} R_{\mathcal{A}} = R_3 &= 3\alpha + \\ &\sum_{r=1}^{\infty}\sum_{s=0}^{\infty} (2+s)G_s^r \alpha^{1+s}(1-\alpha)^{r+s+1} + (3+r+s)F_s^r \alpha^{1+r+s}(1-\alpha)^{r+s} \ , \\ R_{\mathcal{H}} = R_4 &= \sum_{r=1}^{\infty}\sum_{s=0}^{\infty} (2+r)G_s^r \alpha^{1+s}(1-\alpha)^{r+s+1} \ , \end{aligned} \tag{9.33}$$

where the series above can be calculated using series introduced in Lemmas 9.4 and 9.7.

Proof. If assuming $L = 0$ (recall that L denotes the number of consecutive honest blocks proposed after being at state $S_{1,2}$ and before the next adversarial block is proposed), Scenario 2a is equivalent to the event that the block path starting at $(1,2)$ reaches the point $(3,2)$. The probability of this event is α^2, and its corresponding adversarial reward and honest reward are equal to 3 and 0, respectively.

Consider the picture depicted in Figure 9.8. If assuming $L = r$, where $r \geq 1$,

Scenario 2a is equivalent to the event that the block path starting at $(1, 2)$ reaches the point $(1, 2 + r)$, then moves 2 steps to the right to reach the point $(3, 2 + r)$, and then reaches one of the points $\{(3 + r + i, 2 + r + i)|i \geq 0\}$ (line $y = x - 1$) without reaching any of the points $\{(3 + i, 3 + i + r)|i \geq 0\}$ (line $y = x + r$). The probability that the block path starting at $(1, 2)$ reaches the point $(1, 2 + r)$ and then moves to the point $(3, 2 + r)$ is equal to $\alpha^2(1 - \alpha)^r$. Assume the block path reaches the line $y = x - 1$ at point $(3 + r + s, 2 + r + s)$. The event that the block path starting at the point $(3, 2 + r)$ reaches the line $y = x - 1$ for the first time at point $(3 + r + s, 2 + r + s)$ without reaching the line $\{(3 + i, 3 + i + r)|i \geq 0\}$ beforehand is equivalent to the event that that the block path starting at $(0, 0)$ reaches the line $y = x - r$ for the first time at point $(r + s, s)$ without reaching the line $y = x + 1$ beforehand. The probability of the latter event is equal to $F_s^r \alpha^{r+s}(1 - \alpha)^s$, where F_s^r for $r \geq 1$ is introduced in Lemma 9.7. Therefore, the probability that the block path starting at $(1, 2)$ moves to the points $(1, 2 + r)$, and then moves to the point $(3, 2 + r)$, and finally reaches point $(3 + r + s, 2 + r + s)$ before reaching the line $\{(3 + i, 3 + i + r)|i \geq 0\}$ is equal to $F_s^r \alpha^{r+s+2}(1 - \alpha)^{s+r}$. If the block path reaches the line $y = x - 1$ at point $(3 + r + s, 2 + r + s)$, the adversarial reward and the honest reward resulting from Scenario 2a is equal to $3 + r + s$ and 0, respectively. Therefore, the average adversarial reward and honest reward

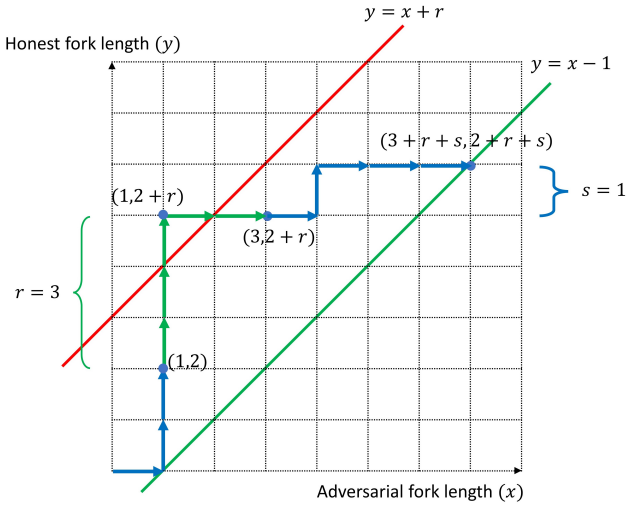


Figure 9.8: Block path in Scenario 2a

under Scenario 2a can be obtained as follows:

$$R'_3 = \frac{3\alpha^2 + \sum_{r=1}^{\infty}\sum_{s=0}^{\infty}(3+r+s)F_s^r\alpha^{r+s+2}(1-\alpha)^{r+s}}{\mathtt{Pr}(\text{Scenario 2a})}\ ,$$
$$R'_4 = 0\ . \tag{9.34}$$

If assuming $L = 0$, Scenario 2b cannot happen.

Consider the picture depicted in Figure 9.9. If assuming $L = r$, where $r \geq 1$, Scenario 2b is equivalent to the event that the block path starting at $(1, 2)$ reaches the point $(1, 2 + r)$, then moves 2 steps to the right to reach the point $(3, 2 + r)$, and then reaches one of the points $\{(3 + i, 3 + i + r)|i \geq 0\}$ (line $y = x + r$) before reaching any of the points $\{(3 + r + i, 2 + r + i)|i \geq 0\}$ (line $y = x - 1$). The probability that the block path starting at $(1, 2)$ reaches the point $(1, 2 + r)$ and then moves to the point $(3, 2 + r)$ is equal to $\alpha^2(1 - \alpha)^r$. Assume the block path reaches the line $y = x + r$ at point $(3 + s, 3 + r + s)$. The event that the block path starting at the point $(3, 2 + r)$ reaches the line $y = x + r$ for the first time at point $(3 + s, 3 + r + s)$ without reaching the line $y = x - 1$ beforehand is equivalent to the event that the block path starting at $(0, 0)$ reaches the line $y = x + 1$ for the first time at point $(s, s + 1)$ without reaching the line $y = x - r$ beforehand. The probability of the latter event is equal to $G_s^r\alpha^s(1 - \alpha)^{s+1}$, where G_s^r for $r \geq 1$ is introduced in Lemma 9.4. Therefore, the probability that the block path starting at $(1, 2)$ moves to the

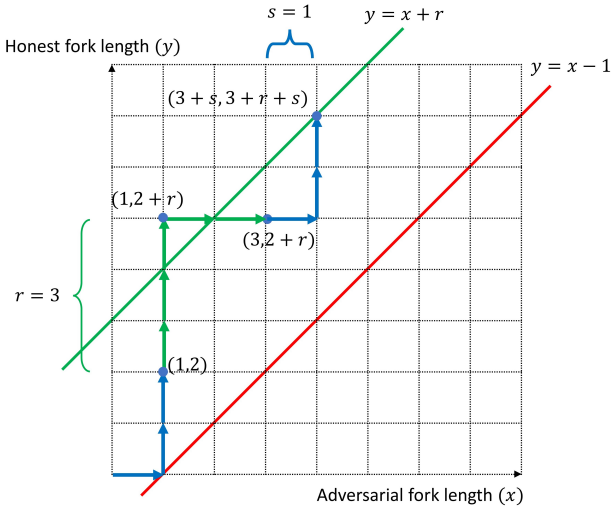


Figure 9.9: Block path in Scenario 2b

point $(1, 2+r)$, and then moves to the point $(3, 2+r)$, and finally reaches the point $(3+s, 3+r+s)$ before reaching the line $y = x - 1$ is equal to $G_s^r \alpha^{s+2}(1-\alpha)^{r+s+1}$. If the block path reaches the line $y = x + r$ at point $(3+s, 3+r+s)$, the adversarial reward and the honest reward resulting from Scenario 2b is equal to $2+s$ and $2+r$, respectively. Therefore, the average adversarial reward and honest reward under Scenario 2b can be obtained as follows:

$$\begin{aligned} R_3'' &= \frac{\sum_{r=1}^{\infty}\sum_{s=0}^{\infty}(2+s)G_s^r\alpha^{2+s}(1-\alpha)^{r+s+1}}{\texttt{Pr}(\text{Scenario 2b})}\ , \\ R_4'' &= \frac{\sum_{r=1}^{\infty}\sum_{s=0}^{\infty}(2+r)G_s^r\alpha^{2+s}(1-\alpha)^{r+s+1}}{\texttt{Pr}(\text{Scenario 2b})}\ . \end{aligned} \tag{9.35}$$

Finally, the average adversarial and honest rewards resulting from the transition from state $S_{1,2}$ to state $S_{0,0}$ under $\texttt{action}_2$ (Scenario 2 occurrence) can be obtained as follows:

$$\begin{aligned} R_{\mathcal{A}} = R_3 &= \frac{\texttt{Pr}(\text{Scenario 2a})R_3' + \texttt{Pr}(\text{Scenario 2b})R_3''}{\texttt{Pr}(\text{Scenario 2})}\ , \\ R_{\mathcal{H}} = R_4 &= \frac{\texttt{Pr}(\text{Scenario 2a})R_4' + \texttt{Pr}(\text{Scenario 2b})R_4''}{\texttt{Pr}(\text{Scenario 2})}\ . \end{aligned} \tag{9.36}$$

As $\texttt{Pr}(\text{Scenario 2}) = \alpha$, the equations presented in 9.33 can be obtained. □

Having access to R_1, R_2, R_3, and R_4, we can use the Markov chain depicted in Figure 9.5, to calculate the block ratio achieved by following strategy $\pi^{\text{S-PR}}$. In Figure 9.10, we compare the block ratio achieved by strategy $\pi^{\text{S-PR}}$ with those achieved by the honest strategy and the optimal PoW selfish mining strategy [22] for an attacker with communication capability $\eta = 0.5$. As can be seen in Figure 9.5, strategy $\pi^{\text{S-PR}}$ can dominate the optimal PoW selfish mining strategy for some range of stake shares.

In Figure 9.11, we compare the block ratio achieved by strategy $\pi^{\text{S-PR}}$ with those achieved by the honest strategy, the optimal PoW selfish mining strategy [22], and the nothing-at-stake selfish mining strategy introduced in [13] (denoted by S-PR2) for an attacker with communication capability $\eta = 0$. Strategy $\pi^{\text{S-PR}}$ can dominate both the optimal PoW selfish mining strategy and the strategy S-PR2 for some range of stake shares.

9.A5 Strategy $\pi^{\texttt{SP1}}$ in Full-Predictable Protocols

We use a two-dimensional (x, y)-grid to depict the chain race. The proposing sequence is represented by a path on this grid, which is referred to as the "block

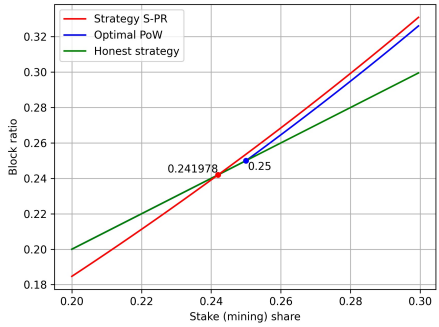


Figure 9.10: Block ratio comparison (communication capability $\eta = 0.5$)

path". Whenever the time slot is honest, i.e., the block proposer of the time slot is honest, the block path moves one step up. Whenever the time slot is adversarial, i.e., the block proposer of the time slot is the attacker, the block path moves one step to the right.

To calculate the block ratio of strategy $\pi^{\mathtt{SP1}}$, we first present two helpful lemmas.

Lemma 9.9. The probability that the block path starting at $(0, 0)$ never reaches the line $y = x$ for $x \geq 1$ is equal to $1 - 2\alpha$.

Proof. To never reach the line $y = x$ for $x \geq 1$, the block path should move one step up to the point $(0, 1)$ as the first step, the probability of which is equal to $1 - \alpha$. We need to calculate the probability of the event that the block path starting at $(0, 1)$ never reaches the line $y = x$ for $x \geq 1$. To this end, we calculate the probability of the complementary event, i.e., the event that the

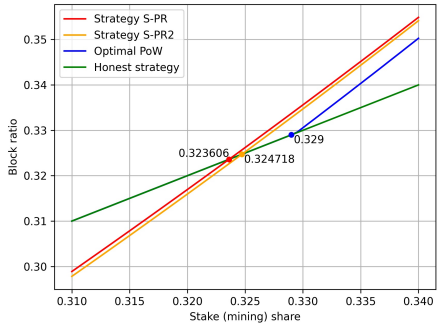


Figure 9.11: Block ratio comparison (communication capability $\eta = 0$)

block path starting at $(0,1)$ reaches the line $y=x$ at least once. Assume the block path starting at $(0,1)$ reaches the line $y=x$ for the first time at point (s,s) for $s \geq 1$. The number of paths from point $(0,1)$ to the point (s,s) for $s \geq 1$ without reaching the line $y=x$ before point (s,s) is equal to $s-1^{\text{th}}$ Catalan number denoted by c_{s-1}. Therefore, the probability of the event that the block path starting at $(0,1)$ reaches the line $y=x$ for the first time at point (s,s) for $s \geq 1$ is equal to $c_{s-1}\alpha^s(1-\alpha)^{s-1}$. As a result, the probability of the event that the block path starting at $(0,1)$ reaches the line $y=x$ at least once can be obtained as follows:

$$\sum_{s=1}^{\infty} c_{s-1}\alpha^s(1-\alpha)^{s-1} = \alpha \sum_{s=0}^{\infty} c_s(\alpha(1-\alpha))^s = \frac{\alpha}{1-\alpha} \ . \tag{9.37}$$

The formula for calculating the series in the equation above is presented in [8]. Therefore, the probability of the event that the block path starting at $(0,1)$ never reaches the line $y=x$ is equal to $1-\frac{\alpha}{1-\alpha} = \frac{1-2\alpha}{1-\alpha}$. Finally, the probability that the block path starting at $(0,0)$ never reaches the line $y=x$ for $x \geq 1$ is equal to $(1-\alpha)\frac{1-2\alpha}{1-\alpha} = 1-2\alpha$. □

Lemma 9.10. The probability that the block path starting at $(0,0)$ reaches the line $y=x-1$ but never passes it is equal to $\frac{\alpha(1-2\alpha)}{(1-\alpha)^2}$.

Proof. The number of paths from point $(0,0)$ to point $(s+1,s)$ without passing the line $y=x-1$ is equal to $s+1^{\text{th}}$ Catalan number denoted by c_{s+1}. Therefore, the probability of the event that the block path starting at point $(0,0)$ reaches the line $y=x-1$ at point $(s+1,s)$ without passing the line $y=x-1$ is equal to $c_{s+1}\alpha^{s+1}(1-\alpha)^s$. The event that the block path starting at point $(s+1,s)$ never reaches the line $y=x-1$ again is equivalent to the event that the block path starting at $(0,0)$ never reaches the line $y=x$ again, the probability of which is presented in Lemma 9.9. Therefore, the probability of the event that the block path starting at point $(0,0)$ reaches the line $y=x-1$ for the last time at point $(s+1,s)$ without ever passing the line $y=x-1$ is equal to $c_{s+1}\alpha^{s+1}(1-\alpha)^s(1-2\alpha)$. Therefore, the probability that the block path starting at $(0,0)$ reaches the line $y=x-1$ but never passes it can be obtained as follows:

$$\begin{aligned} &\sum_{s=0}^{\infty} c_{s+1}\alpha^{s+1}(1-\alpha)^s(1-2\alpha) = \frac{1-2\alpha}{1-\alpha}\sum_{s=1}^{\infty} c_s(\alpha(1-\alpha))^s \\ &= \frac{1-2\alpha}{1-\alpha}\left(\frac{1}{1-\alpha}-1\right) = \frac{\alpha(1-2\alpha)}{(1-\alpha)^2} \ . \end{aligned} \tag{9.38}$$

□

Following strategy SP1, the main chain can be divided into two recurrent cycles: the adversarial cycle and the honest cycle. An adversarial cycle starts at time slot t, where t represents the time slot whose corresponding longest dominant chain is a non-empty set, i.e., $L^{\mathtt{LDC}}(t) > 0$. During adversarial cycles, a set of consecutive adversarial blocks is added to the main chain, and honest blocks get orphaned. Between every two adversarial cycles, there is an honest cycle, where a set of consecutive honest blocks is added to the main chain. Honest cycles include the time slots whose corresponding longest dominant chain is an empty set. Therefore, to obtain the block ratio, the average length of both honest and adversarial cycles should be calculated.

Lemma 9.11. Let $L_{\mathcal{H}}^{\mathtt{SP1}}$ denote the length of the honest cycle $\mathtt{Cycle}^{\mathcal{H}}$ under the strategy SP1. We have:

$$\mathbb{E}(L_{\mathcal{H}}^{\mathtt{SP1}}) = \frac{1+\alpha}{\alpha} \quad . \tag{9.39}$$

Proof. Assume the honest cycle $\mathtt{Cycle}^{\mathcal{H}}$ starts at time slot t from the point $(0,0)$ as depicted in Figure 9.12. The assumption that $\mathtt{Cycle}^{\mathcal{H}}$ starts from point $(0,0)$ results in the block path never reaching the line $y = x$ at $x > 0$. To illustrate this fact, assume the previous adversarial cycle before $\mathtt{Cycle}^{\mathcal{H}}$ starts at time slot t', where $t' < t$, and ends at time slot $t-1$. Assume further that there exists a time slot such as $t'' > t$ at which the block path reaches the line $y = x$ at $x > 0$. In this case, the attacker could win the chain race within the interval $[t', t'']$, which is against our assumption that the chain race within the interval $[t', t-1]$ is the longest dominant chain starting at time slot t'. It is obvious that $L_{\mathcal{H}}^{\mathtt{SP1}}$ cannot be equal to 0 or 1 because in such cases the block path reaches the line $y = x$ at $x > 0$. Therefore, $L_{\mathcal{H}}^{\mathtt{SP1}} \geq 2$.

Knowing that the block path never reaches the line $y = x$ at $x > 0$, $L_{\mathcal{H}}^{\mathtt{SP1}} = i$ represents the event that the block path reaches the line $y = x + (i-1)$ at $x > 0$ without reaching any of the lines $y = x + (j-1)$ at $x > 0$ for $1 < j < i$ given that the block path never reaches the line $y = x$. This means that during the honest cycle, the block path goes i steps up (i consecutive honest blocks are added to the main chain within the interval $[t, t + (i-1)]$) to reach point $(0, i)$, and in the subsequent adversarial cycle which starts at $t + i$, the block path reaches the line $y = x + (i-1)$ at $x > 0$ starting from point $(0, i)$ without never passing it. Reaching the line $y = x + (i-1)$ at $x > 0$ starting from point $(0, i)$ indicates that $\mathtt{LDC}(t+i) > 0$, and thus, slot $t + i$ is the starting slot of a new adversarial cycle. Note that the block path cannot pass the line $y = x + (i-1)$ at $x > 0$ because otherwise there should have been a time slot $t' \in [t, t + (i-1)]$ with $L^{\mathtt{LDC}}(t') > 0$, which means that the honest cycle length should have been less than i. Let E_1 and E_2 represent the event that the block path starting at point $(0,0)$ reaches the line $y = x + (i-1)$ at $x > 0$ without reaching any of

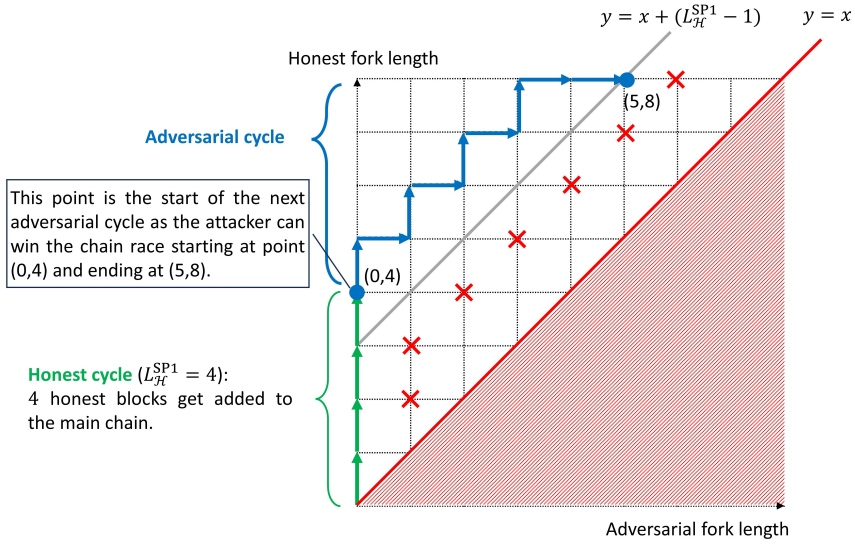


Figure 9.12: Honest cycle of the strategy SP1

the lines $y = x + (j-1)$ at $x > 0$ for $1 < j < i$ and the event that the block path starting at point $(0,0)$ never reaches the line $y = x$ at $x > 0$, respectively.

$$\mathtt{Pr}(L_{\mathcal{H}}^{\mathrm{SP1}} = i) = \mathtt{Pr}(E_1 | E_2) = \frac{\mathtt{Pr}(E_1 \cap E_2)}{\mathtt{Pr}(E_2)} \ . \tag{9.40}$$

Event $E_1 \cap E_2$ represents the event that the block path reaches the line $y = x + (i-1)$ at $x > 0$ without reaching any of the lines $y = x + (j-1)$ at $x > 0$ for $0 < j < i$. The probability that the block path reaches point $(0, i)$ is equal to $(1-\alpha)^i$. The event that the block path starting at point $(0, i)$ reaches the line $y = x + (i-1)$ but never passes it is equivalent to the event that the block path starting at point $(0,0)$ reaches the line $y = x - 1$ but never passes it. The probability of the latter event, which is presented in Lemma 9.10, is equal to $\frac{\alpha(1-2\alpha)}{(1-\alpha)^2}$. Therefore, for $i \geq 2$, we have:

$$\mathtt{Pr}(E_1 \cap E_2) = (1-\alpha)^i \frac{\alpha(1-2\alpha)}{(1-\alpha)^2} \ . \tag{9.41}$$

As according to Lemma 9.9, the probability of event E_2 is equal to $1 - 2\alpha$, $\mathtt{Pr}(L_{\mathcal{H}}^{\mathrm{SP1}} = i)$ for $i \geq 2$ can be obtained as follows:

$$\mathtt{Pr}(L_{\mathcal{H}}^{\mathrm{SP1}} = i) = (1-\alpha)^i \frac{\alpha}{(1-\alpha)^2} \ . \tag{9.42}$$

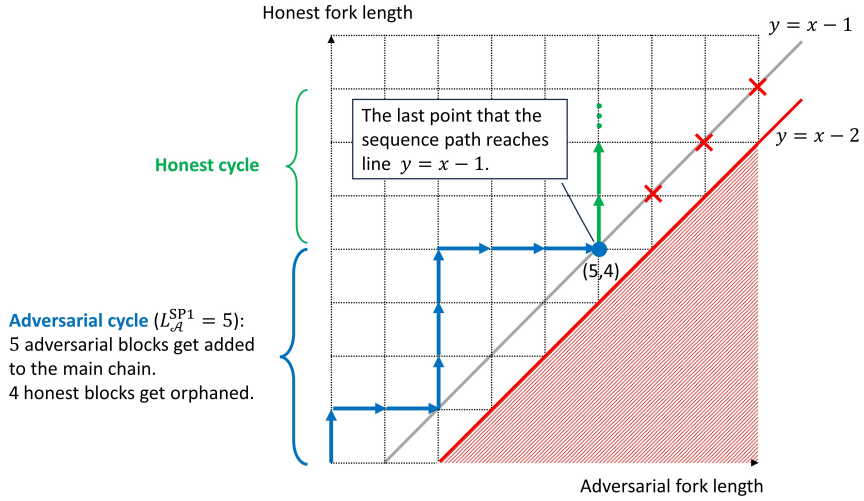


Figure 9.13: Adversarial cycle of the strategy SP1

Therefore, the average length of an honest cycle can be calculated as follows:

$$\mathbb{E}(L_{\mathcal{H}}^{\text{SP1}}) = \sum_{i=2}^{\infty} i \cdot \Pr(L_{\mathcal{H}}^{\text{SP1}} = i) = \frac{1+\alpha}{\alpha} \quad . \tag{9.43}$$

□

Lemma 9.12. Let $L_{\mathcal{A}}^{\text{SP1}}$ denote the length of an adversarial cycle $\texttt{Cycle}^{\mathcal{A}}$ under the strategy SP1. We have:

$$\mathbb{E}(L_{\mathcal{A}}^{\text{SP1}}) = \frac{1}{1-2\alpha} \quad . \tag{9.44}$$

Proof. Assume the adversarial cycle $\texttt{Cycle}^{\mathcal{A}}$ starts at time slot t from the point $(0,0)$ as depicted in Figure 9.13. The assumption that $\texttt{Cycle}^{\mathcal{A}}$ starts at point $(0,0)$ results in the event that the block path starting at $(0,0)$ reaches the line $y = x - 1$ but never passes it. Because, if the block path starting at $(0,0)$ never reaches the line $y = x - 1$, then the $\texttt{LDC}(t)$ would be an empty set, indicating that the time slot t cannot be the starting time slot of an adversarial cycle. Besides, if the block path starting at $(0,0)$ passes the line $y = x - 1$, there should be a time slot $t' < t$ with $L^{\texttt{LDC}}(t') > 0$ within the previous honest cycle. This indicates that the adversarial cycle $\texttt{Cycle}^{\mathcal{A}}$ should start at an earlier time slot rather than t, which is against our assumption.

Knowing that the block path reaches the line $y = x - 1$ but never passes it, $L_{\mathcal{A}}^{\mathtt{SP1}} = i$ represents the event that the block path reaches the line $y = x - 1$ for the last time at point $(i, i-1)$, where $i \geq 1$, given that the block path reaches the line $y = x - 1$ but never passes it. Let E_1 and E_2 represent the event that the block path reaches the line $y = x - 1$ for the last time at point $(i, i-1)$ and the event that the block path reaches the line $y = x - 1$ but never passes it. We have:

$$\mathtt{Pr}(L_{\mathcal{A}}^{\mathtt{SP1}} = i) = \mathtt{Pr}(E_1 | E_2) = \frac{\mathtt{Pr}(E_1 \cap E_2)}{\mathtt{Pr}(E_2)} \ . \tag{9.45}$$

Event $E_1 \cap E_2$ represents the event that the block path reaches the line $y = x-1$ for the last time at point $(i, i-1)$ without ever passing it. According to the proof presented in Lemma 9.10, the probability $\mathtt{Pr}(E_1 \cap E_2)$ can be calculated as follows:

$$\mathtt{Pr}(E_1 \cap E_2) = c_i \alpha^i (1-\alpha)^{i-1}(1-2\alpha) \ . \tag{9.46}$$

Since according to Lemma 9.10, the probability of event E_2 is equal to $\frac{\alpha(1-2\alpha)}{(1-\alpha)^2}$, $\mathtt{Pr}(L_{\mathcal{A}}^{\mathtt{SP1}} = i)$ can be obtained as follows:

$$\mathtt{Pr}(L_{\mathcal{A}}^{\mathtt{SP1}} = i) = \frac{c_i \alpha^i (1-\alpha)^{i-1}(1-2\alpha)}{\frac{\alpha(1-2\alpha)}{(1-\alpha)^2}} \ . \tag{9.47}$$

Therefore, the average length of an adversarial cycle can be obtained as follows:

$$\mathbb{E}(L_{\mathcal{A}}^{\mathtt{SP1}}) = \sum_{i=1}^{\infty} i \cdot \mathtt{Pr}(L_{\mathcal{A}}^{\mathtt{SP1}} = i) = \frac{1}{1-2\alpha} \ . \tag{9.48}$$

Note that during an adversarial cycle with length $L_{\mathcal{A}}^{\mathtt{SP1}} = i$, i adversarial blocks are added to the main chain, and $i - 1$ honest blocks get orphaned. □

Proof of Theorem 9.1. Using Lemmas 9.11 and 9.12, the block ratio of an attacker under strategy $\pi^{\mathtt{SP1}}$ can be obtained as follows:

$$\mathtt{BlkRatio}_{\mathcal{A}}(\pi^{\mathtt{SP1}}) = \frac{\mathbb{E}(L_{\mathcal{A}}^{\mathtt{SP1}})}{\mathbb{E}(L_{\mathcal{A}}^{\mathtt{SP1}}) + \mathbb{E}(L_{\mathcal{H}}^{\mathtt{SP1}})} = \frac{\alpha}{1-2\alpha^2} \ . \tag{9.49}$$

□

9.A6 Strategy $\pi^{\mathtt{SP2}}$ in Full-Predictable Protocols

To calculate the block ratio of strategy $\pi^{\mathtt{SP2}}$, we first present a couple of helpful lemmas.

Lemma 9.13. If assuming the number of paths in a (x, y)-grid from start point $(0, 0)$ to the point (s, s) without reaching the line $y = x - r$ for $r \geq 1$ is denoted by D_s^r (define D_0^r to be equal to 1), then we have:

$$\begin{aligned}
&\sum_{s=0}^{\infty} D_s^r \big(\alpha(1-\alpha)\big)^s = \frac{1}{1-2a}\Big(1 - (\frac{\alpha}{1-\alpha})^r\Big) \ , \\
&\sum_{s=0}^{\infty} sD_s^r \big(\alpha(1-\alpha)\big)^s = \frac{2\alpha(1-\alpha)}{(1-2\alpha)^3} \\
&- \Big(\frac{\alpha}{1-\alpha}\Big)^r \Big(\frac{2\alpha(1-\alpha) + r(1-2\alpha)}{(1-2\alpha)^2}\Big) \ .
\end{aligned} \tag{9.50}$$

Proof. Let $r \geq 1$. The probability that a block path starting at $(0, 0)$ reaches the point (s, s) without reaching the line $y = x - r$ is equal to $D_s^r \big(\alpha(1-\alpha)\big)^s$. Therefore, the probability that a block path starting at $(0, 0)$ reaches the line $y = x$ for the last time at the point (s, s) without ever reaching the line $y = x - r$ is equal to $P_s^r = D_s^r(1-2\alpha)\big(\alpha(1-\alpha)\big)^s$. The sum of all the probabilities P_s^r for $s \in \mathbb{W}$ is the same as the probability that a block path never reaches the line $y = x - r$. Therefore, using Lemma 9.3, we obtain:

$$\begin{aligned}
&\sum_{s=0}^{\infty} D_s^r (1-2\alpha)\big(\alpha(1-\alpha)\big)^s = 1 - (\frac{\alpha}{1-\alpha})^r \Rightarrow \\
&\sum_{s=0}^{\infty} D_s^r \big(\alpha(1-\alpha)\big)^s = \frac{1}{1-2a}\Big(1 - (\frac{\alpha}{1-\alpha})^r\Big) \ .
\end{aligned} \tag{9.51}$$

To prove the second equality in Lemma 9.13, we use the variable substitution $\alpha(1-\alpha) = x$ in the equality above. We have:

$$\sum_{s=0}^{\infty} D_s^r x^s = \frac{1 - \Big(\frac{1-\sqrt{1-4x}}{1+\sqrt{1-4x}}\Big)^r}{\sqrt{1-4x}} \ . \tag{9.52}$$

By taking the derivative from both sides, we obtain:

$$\begin{aligned}
&\sum_{s=0}^{\infty} sD_s^r x^{s-1} = \frac{2}{\sqrt{(1-4x)^3}} + \\
&\frac{4(1-\sqrt{1-4x})^{r-1}\Big(8x^2 - 2x - r\sqrt{(1-4x)^3}\Big)}{(1+\sqrt{1-4x})^{r+1}\sqrt{(1-4x)^5}} \ .
\end{aligned} \tag{9.53}$$

By multiplying both sides to x and substituting $x = \alpha(1-\alpha)$, the second equality in Lemma 9.13 can be obtained. □

Following strategy SP2, the main chain can be divided into two recurrent cycles: the adversarial cycle and the honest cycle. Note that in strategy SP2, an adversarial cycle always starts at an adversarial time slot. To obtain the block ratio of strategy SP2, we first calculate the average length of both honest and adversarial cycles.

Lemma 9.14. Let $L_{\mathcal{H}}^{\text{SP2}}$ denote the length of the honest cycle $\texttt{Cycle}^{\mathcal{H}}$ under the strategy SP2. We have:

$$\mathbb{E}(L_{\mathcal{H}}^{\text{SP2}}) = \frac{1}{\alpha(1-\alpha)} \quad . \tag{9.54}$$

Proof. Assume the honest cycle $\texttt{Cycle}^{\mathcal{H}}$ starts at time slot t from the point $(0,0)$ as depicted in Figure 9.14. The assumption that $\texttt{Cycle}^{\mathcal{H}}$ starts from the point $(0,0)$ results in the event that the block path starting at point $(0,0)$ never reaches the line $y = x$ at $x > 0$. The reason is the same as the explanation presented in Lemma 9.11. It is obvious that $L_{\mathcal{H}}^{\text{SP2}}$ cannot be equal to 0 or 1 because in such cases, the block path reaches the line $y = x$ at $x > 0$. Therefore, $L_{\mathcal{H}}^{\text{SP2}} \geq 2$.

Knowing that the block path never reaches the line $y = x$ at $x > 0$, $L_{\mathcal{H}}^{\text{SP2}} = i$ represents the event that the block path reaches point $(0, i)$ and then moves to point $(1, i)$ given that it never reaches the line $y = x$ at $x > 0$. It means that

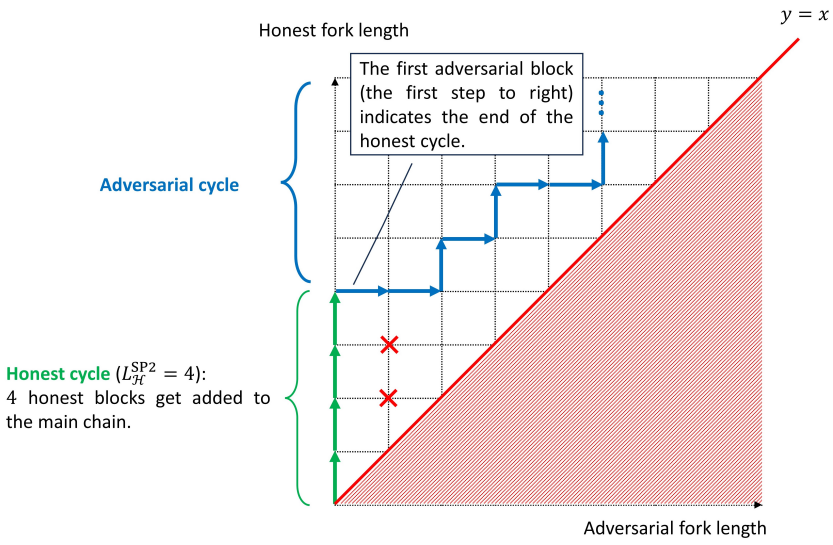


Figure 9.14: Honest cycle of the strategy SP2

during the honest cycle, the block path moves i steps up to reach point $(0, i)$ (i consecutive honest blocks are added to the main chain within the interval $[t, t + (i - 1)]$), and then, the block path moves one step right at time slot $t + i$ to reach point $(1, i)$. This indicates that time slot $t + i$ is the starting time slot of the subsequent adversarial cycle. Let E_1 and E_2 represent the event that the block path starting at point $(0, 0)$ reaches point $(0, i)$ and then point $(1, i)$ and the event that the block path starting at point $(0, 0)$ never reaches the line $y = x$ at $x > 0$, respectively.

$$\mathtt{Pr}(L_{\mathcal{H}}^{\text{SP2}} = i) = \mathtt{Pr}(E_1 | E_2) = \frac{\mathtt{Pr}(E_1 \cap E_2)}{\mathtt{Pr}(E_2)} \ . \tag{9.55}$$

Event $E_1 \cap E_2$ represents the event that the block path reaches point $(0, i)$ and then point $(1, i)$ without ever reaching the line $y = x$ at any future points with $x > 0$. The probability that the block path reaches point $(0, i)$ and then point $(1, i)$ is equal to $(1 - \alpha)^i \alpha$. The event that the block path starting at point $(1, i)$ never reaches the line $y = x$ at $x > 0$ is equivalent to the event of never reaching the line $y = x - (i - 1)$ starting at point $(0, 0)$. The probability of the latter event, which is presented in Lemma 9.3, is equal to $1 - (\frac{\alpha}{1-\alpha})^{i-1}$. For $i \geq 2$, we have:

$$\mathtt{Pr}(E_1 \cap E_2) = (1 - \alpha)^i \alpha \Big(1 - \big(\frac{\alpha}{1 - \alpha}\big)^{i-1}\Big) \ . \tag{9.56}$$

Since, according to Lemma 9.9, the probability of event E_2 is equal to $1 - 2\alpha$, $\mathtt{Pr}(L_{\mathcal{H}}^{\text{SP2}} = i)$ for $i \geq 2$ can be obtained as follows:

$$\mathtt{Pr}(L_{\mathcal{H}}^{\text{SP2}} = i) = \frac{(1 - \alpha)^i \alpha \Big(1 - \big(\frac{\alpha}{1-\alpha}\big)^{i-1}\Big)}{1 - 2\alpha} \ . \tag{9.57}$$

Therefore, the average length of an honest cycle can be calculated as follows:

$$\mathbb{E}(L_{\mathcal{H}}^{\text{SP2}}) = \sum_{i=2}^{\infty} i \cdot \mathtt{Pr}(L_{\mathcal{H}}^{\text{SP2}} = i) = \frac{1}{\alpha(1 - \alpha)} \ . \tag{9.58}$$

□

Lemma 9.15. Let $L_{\mathcal{A}}^{\text{SP2}}$ denote the length of an adversarial cycle under strategy SP2. We have:

$$\mathbb{E}(L_{\mathcal{A}}^{\text{SP2}}) = \frac{1 - \alpha(1 - \alpha)}{(1 - \alpha)(1 - 2\alpha)} \ . \tag{9.59}$$

Proof. Assume the adversarial cycle $\mathtt{Cycle}_{\mathtt{new}}^{\mathcal{A}}$ starts at time slot t from the point $(0, 0)$ as depicted in Figure 9.15. Assume further that before $\mathtt{Cycle}_{\mathtt{new}}^{\mathcal{A}}$, there exists an honest cycle $\mathtt{Cycle}^{\mathcal{H}}$ whose length is equal to k. Since the adversarial

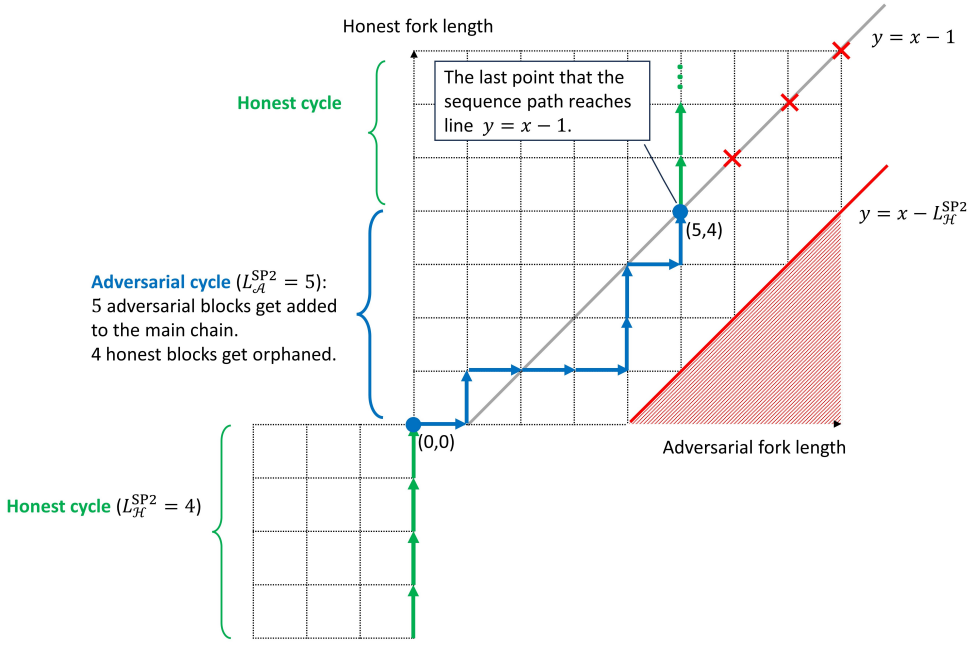


Figure 9.15: Adversarial cycle of the strategy SP2

cycle always starts with a time slot whose proposer is adversarial, the block path always moves one step to the right to reach the point $(1, 0)$. The assumptions that $\texttt{Cycle}_{\texttt{new}}^{\mathcal{A}}$ starts from point $(0, 0)$ and the length of the previous honest cycle is equal to k result in the event that the block path never reaches the line $y = x - k$. To illustrate this fact, assume that before $\texttt{Cycle}^{\mathcal{H}}$, there exists another adversarial cycle $\texttt{Cycle}_{\texttt{old}}^{\mathcal{A}}$ that starts at time slot t_1 and ends at t_2. Assume further that there exists a time slot such as $t_3 > t$ at which the block path reaches the line $y = x - k$. In this case, the attacker can win the chain race within the interval $[t_1, t_3]$, which is against our assumption that the chain race within the interval $[t_1, t_2]$ is the longest dominant chain starting at t_1.

Knowing that the block path never reaches the line $y = x-k$, $L_{\mathcal{A}}^{\mathrm{SP2}} = i$ represents the event that the block path reaches the line $y = x - 1$ for the last time at point $(i, i-1)$ given that the block path never reaches the line $y = x-k$. Let E_1 and E_2 represent the event that the block path starting at point $(1, 0)$ reaches the line $y = x - 1$ for the last time at point $(i, i - 1)$ and the event that the block path starting at point $(1, 0)$ never reaches the line $y = x - k$. We have:

$$\texttt{Pr}(L_{\mathcal{A}}^{\mathrm{SP2}} = i) = \texttt{Pr}(E_1 | E_2) = \frac{\texttt{Pr}(E_1 \cap E_2)}{\texttt{Pr}(E_2)} \quad . \tag{9.60}$$

Event $E_1 \cap E_2$ represents the event that the block path reaches the line $y = x-1$ for the last time at point $(i, i-1)$ without ever reaching the line $y = x-k$. The number of paths from point $(1,0)$ to point $(i, i-1)$ without passing through the line $y = x-k$ is equal to D_{i-1}^{k-1} as introduced in Lemma 9.13. Therefore, the probability of the event that the block path starting at point $(1,0)$ reaches the point $(i, i-1)$ without ever passing the line $y = x-k$ is equal to $D_{i-1}^{k-1}\big(\alpha(1-\alpha)\big)^{i-1}$. To ensure that the block path reaches the line $y = x-1$ for the last time at point $(i, i-1)$, the block path should never reach the line $y = x-1$ again starting from point $(i, i-1)$, the probability of which is equal to $1-2\alpha$. We have:

$$\texttt{Pr}(E_1 \cap E_2) = D_{i-1}^{k-1}\big(\alpha(1-\alpha)\big)^{i-1}(1-2\alpha) \ . \tag{9.61}$$

Since according to Lemma 9.3, the probability of event E_2 is equal to $1 - \big(\frac{\alpha}{1-\alpha}\big)^{k-1}$, $\texttt{Pr}(L_{\mathcal{A}}^{\texttt{SP2}} = i)$ can be obtained as follows:

$$\texttt{Pr}(L_{\mathcal{A}}^{\texttt{SP2}} = i) = \frac{D_{i-1}^{k-1}\big(\alpha(1-\alpha)\big)^{i-1}(1-2\alpha)}{1-\big(\frac{\alpha}{1-\alpha}\big)^{k-1}} \ . \tag{9.62}$$

Therefore, using Lemma 9.13, the average length of an adversarial cycle can be obtained as follows:

$$\mathbb{E}(L_{\mathcal{A}}^{\texttt{SP2}}) = \sum_{i=1}^{\infty} i \cdot \texttt{Pr}(L_{\mathcal{A}}^{\texttt{SP1}} = i) = \frac{1-\alpha(1-\alpha)}{(1-\alpha)(1-2\alpha)} \ . \tag{9.63}$$

Note that during an adversarial cycle with length $L_{\mathcal{A}}^{\texttt{SP2}} = i$, i adversarial blocks are added to the main chain, and $i-1$ honest blocks get orphaned. □

Proof of Theorem 9.2. Using Lemmas 9.14 and 9.15, the block ratio of an attacker under strategy $\pi^{\texttt{SP1}}$ can be obtained as follows:

$$\texttt{BlkRatio}_{\mathcal{A}}(\pi^{\texttt{SP2}}) = \frac{\mathbb{E}(L_{\mathcal{A}}^{\texttt{SP2}})}{\mathbb{E}(L_{\mathcal{A}}^{\texttt{SP2}}) + \mathbb{E}(L_{\mathcal{H}}^{\texttt{SP2}})} = \frac{\alpha(1-\alpha(1-\alpha))}{(1-\alpha)^2(1+\alpha)} \ . \tag{9.64}$$

□

9.A7 Strategy $\pi^{\texttt{0-SP}}$

The algorithm used in the description of strategy $\pi^{\texttt{0-SP}}$ is presented in Algorithm 2.

Algorithm 2 Optimal selfish proposing strategy in full-predictable LC-PoS environments for attacker with communication capability $\eta = 0$

Input: Seq ▷ Seq is a list of -1 and 1 that represents the proposing sequence within the range (t_i, t_{i+1}), where t_i and t_{i+1} are two consecutive checkpoint slots. The adversarial (honest) slot is denoted by 1 (-1).

Output: $leastCycle$ ▷ $leastCycle$ is a list that contains the adversarial slots that serve as the starting points of the adversarial cycles within the range (t_i, t_{i+1}).

$indx \leftarrow 1$
for $t \leftarrow 1$ to $\texttt{len}(Seq)$ **do**
    $H[t] \leftarrow H[t-1] + Seq[t]$ ▷ $H[t]$ represents the height of a block that is proposed at time slot t.
    if $Seq[t] == 1$ **then**
        $aSlots[indx] = t$ ▷ $aSlots$ is a list of adversarial time slots.
        $indx \leftarrow indx + 1$
    end if
end for
$N \leftarrow \texttt{len}(aSlots)$
$Chain \leftarrow$ an empty list
for $indx \leftarrow 1$ to N **do**
    $Chain_{indx} \leftarrow$ an empty list
end for
for $indx \leftarrow 1$ to $N - 1$ **do**
    if $indx == 1$ or $Chain_{indx} \neq \emptyset$ **then**
        $leastCycle_{indx} \leftarrow$ the list with the least number of adversarial cycles in $Chain_{indx}$
        $newLeastCycle_{indx} \leftarrow \texttt{concatenate}(leastCycle_{indx}, aSlots[indx])$
        for $indx' \leftarrow indx$ to $N - 1$ **do**
            if $H[aSlots[indx]] \leq H[aSlots[indx']]$ and $H[aSlots[indx]] > H[aSlots[indx'+1]]$ **then**
                $Chain[indx'+1].\texttt{append}(newLeastCycle_{indx})$
            end if
        end for
        if $H[aSlots[indx]] \leq H[aSlots[N]]$ **then**
            $Chain.\texttt{append}(newLeastCycle_{indx})$
        end if
    end if
end for
if $Chain_N \neq \emptyset$ **then**
    $leastCycle_N \leftarrow$ the list with the least number of adversarial cycles in $Chain_N$
    $newLeastCycle_N \leftarrow \texttt{concatenate}(leastCycle_N, aSlots[N])$
    $Chain.\texttt{append}(newLeastCycle_N)$
end if
$leastCycle \leftarrow$ the list with the least number of adversarial cycles in $Chain$

9.A8 Justifying the Optimality of Strategy $\pi^{0\text{-SP}}$

Let t_i and t_{i+1} be two consecutive checkpoint slots. We first discuss how to find the optimal selfish proposing strategy within the interval (t_i, t_{i+1}) and then prove that this optimal strategy is independent of the strategy that the attacker follows outside the interval (t_i, t_{i+1}). Assume $N^{\mathcal{A}}$ and $N^{\mathcal{H}}$ represent the number of adversarial blocks and the number of honest blocks within the interval (t_i, t_{i+1}), respectively. Note that $N^{\mathcal{H}} \geq N^{\mathcal{A}} - 1$ since otherwise t_{i+1} could not be a checkpoint slot. When following the selfish proposing attack within the interval (t_i, t_{i+1}), not all the $N^{\mathcal{A}} + N^{\mathcal{H}}$ get added to the main chain due to the orphan occurrence. We want to calculate the number of adversarial and honest blocks added to the main chain under the optimal strategy. Due to the proposing sequence predictability in a full-predictable protocol, the attacker can publish his blocks in a way that none of them get orphaned. Therefore, all the $N^{\mathcal{A}}$ adversarial blocks within interval (t_i, t_{i+1}) can get added to the main chain. To calculate the number of min-chain honest blocks, we first define the term "adversarial cycle". An adversarial cycle is a set of $n \geq 1$ consecutive adversarial blocks in the main chain sandwiched by two honest main-chain blocks. Each pair of two consecutive adversarial cycles in the main chain are disconnected by one or more honest blocks. If assuming that the attacker's communication capability is equal to zero, each adversarial cycle comprising n adversarial blocks can orphan at most $n - 1$ honest blocks. The number of honest blocks added to the main chain during the selfish proposing attack depends on the number of adversarial cycles. Assume that c denotes the number of adversarial cycles within the interval (t_i, t_{i+1}). In this case, the number of honest blocks added to the main chain within the interval (t_i, t_{i+1}) is equal to $N^{\mathcal{H}} - N^{\mathcal{A}} + c$. An optimal strategy minimizes the number of main-chain honest blocks. Therefore, to obtain the optimal strategy, we should find a strategy that minimizes the number of adversarial cycles. The method presented in Algorithm 2 brute-forces all the possible combinations of adversarial cycles between two checkpoint slots to find a combination with the lowest number of adversarial cycles.

Assume $\pi_i^{0\text{-SP}}$ is the optimal strategy corresponding to interval (t_i, t_{i+1}), which is obtained from Algorithm 2. Assume further that the number of orphaned honest blocks within interval (t_i, t_{i+1}) under strategy $\pi_i^{0\text{-SP}}$ is equal to $N^{\mathcal{A}} - c$, where c represent the minimum possible number of adversarial cycles within interval (t_i, t_{i+1}). In this part, we discuss knowing about the proposing sequences in intervals $[0, t_i]$ and $[t_{i+1}, \infty)$ does not change the optimal strategy $\pi_i^{0\text{-SP}}$ within interval (t_i, t_{i+1}). To prove this claim we need to show that the adversarial blocks within interval (t_i, t_{i+1}) cannot orphan more than $N^{\mathcal{A}} - c$ honest blocks within and outside interval (t_i, t_{i+1}). Assume that there is an honest block $B^{\mathcal{H}}$ proposed at t', where $t' \notin (t_i, t_{i+1})$. We want to prove that either it is infeasible

for the attacker to orphan block $B^{\mathcal{H}}$ using his adversarial blocks within interval (t_i, t_{i+1}), or if he does succeed in orphaning block $B^{\mathcal{H}}$ using his adversarial blocks within interval (t_i, t_{i+1}), the number of orphaned honest blocks will not increase. Consider the scenario that $t' \in [t_{i+1}, \infty)$. If adversarial blocks within interval (t_i, t_{i+1}) can orphan block $B^{\mathcal{H}}$, they can also orphan the honest block proposed at checkpoint time slot t_{i+1}, which is contradictory to the definition of checkpoint slots. Now consider the scenario that $t' \in [0, t_i]$. In this case, block $B^{\mathcal{H}}$ may get orphaned by the adversarial blocks within interval (t_i, t_{i+1}). Assume the attacker orphans block $B^{\mathcal{H}}$ by an adversarial fork denoted by f, where f contains at least one adversarial block within interval (t_i, t_{i+1}). We claim that if f contains one adversarial block within interval (t_i, t_{i+1}), then all the other adversarial blocks in fork f also belong to interval (t_i, t_{i+1}). To justify this claim, assume fork f contains one adversarial block within interval $[0, t_i]$. In this case, an adversarial fork starting at a time slot earlier than t_i and ending at a time slot later than t_i can orphan an honest block proposed at $t' \leq t_i$, which is contradictory to the assumption that t_i is a checkpoint slot. Knowing that f only contains adversarial blocks within interval (t_i, t_{i+1}), we can prove that modifying strategy $\pi_i^{\texttt{0-SP}}$ to generate fork f cannot result in a greater number of orphaned honest blocks. By following strategy $\pi_i^{\texttt{0-SP}}$, the minimum number of adversarial cycles within interval (t_i, t_{i+1}) is equal to c. Orphaning the honest block proposed at $t' \leq t_i$ using $N^{\mathcal{A}}$ adversarial blocks within interval (t_i, t_{i+1}) leads to the outcome where the number of adversarial cycles either stay the same or becomes greater than c. This is because the interval $[t', t_{i+1})$ contains a greater number of honest blocks compared to interval (t_i, t_{i+1}), while the number of adversarial blocks is the same in both intervals. Therefore, orphaning honest blocks outside interval (t_i, t_{i+1}) using the adversarial blocks within interval (t_i, t_{i+1}) cannot lead to a greater block ratio.

9.A9 Deep Q-Learning

9.A9.1 Implementation Details

In our implementation, the target and Q neural networks are structured with 3 hidden layers, comprising 128, 128, and 64 neurons each. We assigned $k_1 = 5$, indicating that each state stores information regarding the current fork as well as the possible forks that can be generated on top of the 5 most recent honest blocks. Therefore, the total number of actions and neurons in the output layer is equal to 18. In the full-predictable implementation, we set $k_2 = 54$, indicating that the attacker can predict the block proposers of upcoming 54 slots. In the semi-predictable implementation, we set $k_3 = 54$, indicating that the attacker is aware of the time slot for up to 54 future adversarial blocks. Therefore,

in LC-PoS implementations with perfect randomness, full predictability, and semi-predictability, the number of neurons in the input layer is 10, 64, and 64, respectively. In semi-predictable LC-PoS implementation, we set the average block generation rate to be equal to 1 block per 20 slots. We set that if during the training process, $l_{\mathcal{H}}$ becomes greater than $l_{\mathcal{A}} + 5$, the action "`wait` in the current fork" gets removed from the possible action set. This is to speed up the training process. A greater bound may result in a more profitable strategy; however, it reduces the training speed. The results shown in Figures 9.4a and 9.4b are obtained by taking the average of the block ratio achieved over 1,000,000 steps of state transitions under the trained neural networks.

9.A9.2 DQL Implementation Considering Node Locations

To obtain the optimal selfish proposing strategy, a proposer should be aware of his stake share and communication capability. In PoS protocols, it is almost straightforward for the proposers to calculate their stake share using account information available in the blockchain ledger. This shows that proposers have a good understanding of their stake shares. However, the concept of communication capability is a bit unclear since it is not straightforward to specify the number of nodes that receive the selfish proposer's block first during a block race. To move towards a more practical implementation, rather than passing communication capability as an input parameter to the environment, we can implement an environment that encompasses the location of proposer nodes.

We implement the network according to the model introduced in [14]. We assume that the proposer nodes are distributed according to the Poisson point process within a spatial disk. Let $d_{i,j}$ and $cd_{i,j}$ denote the distance and the communication delay between proposer nodes P_i and P_j in the spatial disk. We assume further that the communication delay between proposer nodes P_i and P_j follows a normal distribution with mean $\mu = k \cdot d_{i,j}$, which is proportional to the distance between the nodes P_i and P_j, and a constant variance σ^2. Let $P^{\mathcal{A}}$ denote the set of adversarial nodes. Consider the scenario in which honest proposer $P_i^{\mathcal{H}}$ and the attacker simultaneously publish two new blocks to the network. Assume honest proposer $P_j^{\mathcal{H}}$ is the proposer of the upcoming time slot. The attacker wins the block race if there exists at least one adversarial proposer $P_k^{\mathcal{A}} \in P^{\mathcal{A}}$ that can satisfy the following condition:

$$cd_{i,k} + cd_{k,j} \leq cd_{i,j} \quad . \tag{9.65}$$

The condition represented above implies that the proposer of the next slot receives the adversarial block earlier than the honest block and, consequently, proposes his new block on top of the adversarial block. We trained the DQL agent

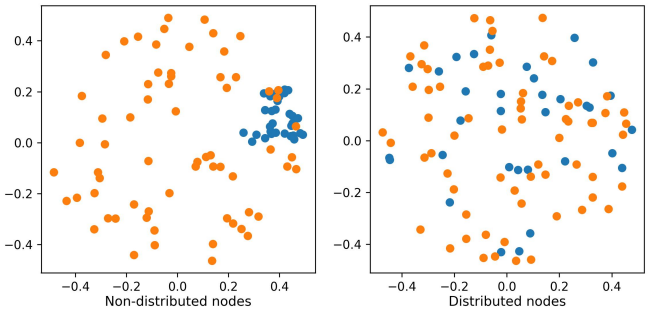


Figure 9.16: Node locations

for an attacker with stake share $\alpha = \frac{1}{3}$ under two different network conditions depicted in Figure 9.16: the distributed model and the non-distributed model. In the distributed model, the adversarial nodes are well-distributed among the honest nodes. Conversely, in the non-distributed model, the adversarial nodes are almost highly concentrated. In our implementation, we assigned $k = 1$ and $\sigma = \frac{1}{25}$. The block ratios achieved under the distributed model and the non-distributed model are equal to 0.4466 and 0.4051, respectively. This demonstrates that a high degree of node distribution can lead to an increase in the attacker's block ratio.

Chapter 10

Commitment Attacks on Ethereum's Reward Mechanism

Publication Data

Roozbeh Sarenche, Ertem Nusret Tas, Barnabé Monnot, Caspar Schwarz-Schilling, and Bart Preneel, "Commitment Attacks on Ethereum's Reward Mechanism," in *European Symposium on Security and Privacy (EuroS&P)*, pages 504–526, IEEE, 2025 (https://arxiv.org/abs/2407.19479).

Contributions

I was one of the two main authors involved in Conceptualization, Methodology, Writing Original Draft, Software, and Visualization. Ertem Nusret Tas and I contributed equally and are listed alphabetically.

Abstract. Validators in permissionless, large-scale blockchains, such as Ethereum, are typically payoff-maximizing, *rational* actors. Ethereum relies on in-protocol incentives, like rewards for correct and timely votes, to induce *honest* behavior and secure the blockchain. However, external incentives, such as the block proposer's opportunity to capture maximal extractable value (MEV), may tempt validators to deviate from honest protocol participation.

We show a series of commitment attacks on LMD GHOST, a core part of Ethereum's consensus mechanism. We demonstrate how a single adversarial block proposer can orchestrate long-range chain reorganizations by manipulating Ethereum's reward system for timely votes. These attacks disrupt the intended balance of power between proposers and voters: by leveraging credible threats, the adversarial proposer can coerce voters from previous slots into supporting blocks that conflict with the honest chain, enabling a chain reorganization.

In response, we introduce a novel reward mechanism that restores the voters' role as a check against proposer power. Our proposed mitigation is fairer and more decentralized – not only in the context of these attacks – but also practical for implementation in Ethereum.

10.1 Introduction

10.1.1 Ethereum Consensus Security

Blockchains have emerged as one of the most prominent applications of distributed consensus, a field over 40 years old. Whereas earlier blockchains (*e.g.*, Bitcoin) used proof-of-work (PoW) based protocols, the last few years witnessed a proliferation of proof-of-stake (PoS) protocols, where financial stake rather than compute power determines the protocol participants. The largest proof-of-stake (PoS) blockchain by market capitalization is Ethereum, secured by a PoS Byzantine fault tolerant (BFT) consensus protocol (Gasper [34]). Ethereum's consensus is run by a set of *validators*, which output *two* ledgers: the *available ledger* and the *finalized ledger*. Requirements for these ledgers were formally captured by the *ebb-and-flow* property [59]: (i) The *available ledger* must remain secure (*i.e.*, safe and live) under a synchronous network despite unexpected and large drops in the number of active validators (this is also known as the sleepy network model [63], or dynamic availability [27]). (ii) The *finalized ledger* must satisfy *accountable safety*[1] under a partially synchronous network and must stay live if sufficiently many validators are active. Finally, (iii) the finalized ledger must always be a *prefix* of the available ledger. To

[1] Namely, if safety is ever violated, the honest validators can identify the adversarial parties responsible for the violation (see Appendix 10.A1.2).

achieve the ebb-and-flow property, Ethereum employs two sub-protocols: LMD GHOST (Latest Message Driven Greedy Heaviest Observed Sub-Tree) that outputs a *canonical chain* as the available ledger, and Casper FFG (the Friendly Finality Gadget [33]) that checkpoints blocks on this canonical chain. These checkpointed blocks constitute the finalized ledger.

Ethereum's consensus has been subjected to much scrutiny, as evidenced by attacks on the security of both LMD GHOST [56, 58–60, 69] and Casper FFG [53, 54] as well as various protocol patches proposed as response (see [77] for a comprehensive history). Without the safety of the canonical chain (available ledger), Casper FFG would fail to checkpoint blocks, causing the finalized ledger to stall indefinitely [35]. Similarly, without the liveness of the canonical chain, Casper FFG cannot ensure the liveness of the finalized ledger. Thus, the security of LMD GHOST that outputs the canonical chain is crucial for the security of both ledgers.

Many Ethereum users today act on time-sensitive transactions as soon as they enter the canonical chain, without waiting the 12.8 minutes required for their finalization by Casper FFG, or any other notion of confirmation by LMD GHOST. This makes them susceptible to instability on the chain tip. Making matters worse, validators are incentivized to reorganize (*reorg*) the blocks at the end of the canonical chain by proposing new blocks on top of older blocks rather than the chain tip. The main reason for this is the magnitude of maximal extractable value (MEV) [19], *i.e.*, the payoff of a block proposer due to its control over the inclusion, exclusion, and ordering of transactions (*e.g.*, the proposer can frontrun arbitrage transactions in trades, backrun liquidation events to buy the liquidated assets) [38]. Once the blocks at the chain tip are replaced (*i.e.*, *reorged*), the new blocks can re-include the transactions of the reorged blocks, and capture the associated MEV. Therefore, besides security, preventing reorgs emerged as a goal of Ethereum consensus, leading to new reorg-resilient proposals to replace LMD GHOST (*e.g.*, Goldfish [39], RLMD-GHOST [41]).

10.1.2 Incentives in Ethereum

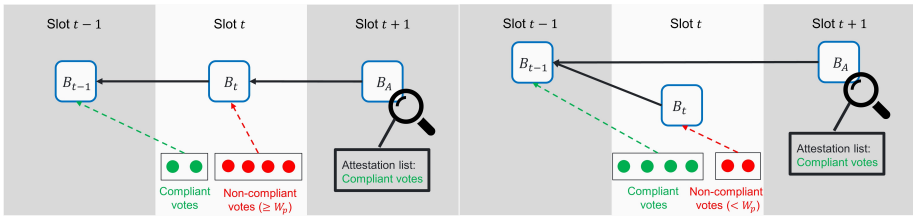
LMD GHOST proceeds in slots of 12 seconds, each with a unique block proposer (hereafter called the *leader*) that proposes a new block extending the tip of the canonical chain. Each slot is also assigned a committee of validators called *attestors* that vote for the block at the tip of the canonical chain in their views. To deter future leaders from reorging blocks, these attestors must be incentivized to vote for the proposal at the canonical chain's tip before the next slot starts; so that the proposed block gathers enough support against any future competitor. Hence, incentivizing the timeliness of these votes is crucial

for mitigating reorg attempts and achieving a notion of reorg-resilience in the presence of payoff-maximizing (*i.e.*, *rational*) validators, which might otherwise find it profitable to delay their votes or vote for old blocks. For this purpose, Ethereum uses a so-called *head vote reward mechanism*: a vote sent for some slot t is rewarded on the canonical chain only if the vote is included in a block B proposed at slot $t+1$ (*i.e.*, *timely*) and if the vote is for the last block from slots $[0, t]$ on the canonical chain (*i.e.*, *correct*). Ethereum also implements an *inclusion reward mechanism*: a slot $t+1$ leader is rewarded for including the (timely) slot t votes in its block. Opposing these rewards, future leaders are motivated by MEV to explore ways of deterring previous slots' attestors from sending timely votes. The tension between the opposing incentives has so far resisted resolution in the form of an attack or a security proof considering rational validators.

10.1.3 Commitment Attacks on LMD GHOST

In this work, we present a series of attacks on the reorg-resilience and security of the latest version of LMD GHOST [77]. Their scope ranges from reorging a single block to violating the security of LMD GHOST (*i.e.*, the available ledger), which in turn causes a liveness violation for Casper FFG's finalized ledger. The attacks exploit the head vote reward mechanism in the presence of rational attestors. Recall that the attestor sending a slot t head vote is rewarded only if the vote is included in a block of slot $t+1$. The core idea behind our attacks is that an adversarial leader can abuse its power over the reward of the previous slot's attestors to incentivize them to vote for blocks favorable to the adversary.

The attacks can be conducted by even a single adversarial validator who owns the minimum amount of stake. However, the greater the number of validators controlled by the adversary, the more frequently the attacks can be executed. The adversarial strategy involves only *committing* to a credible threat, which the attestors must be aware of to react rationally (see Section 10.4.3 for details). In today's Ethereum, over 90% of the validators run an out-of-protocol software, 'mev boost', in which block proposers outsource the building of their block content to specialized third parties [20, 74]. It is easily imaginable to extend this widely distributed software program to also facilitate the communication between the adversary and attestors in our attacks. Then, as long as the majority of the attestors are rational (and the remaining ones can be correct), the attacks are guaranteed to be successful.



(a) Unsuccessful attack (b) Successful attack

Figure 10.1: The simple game in the presence of non-colluding (solo) validators. As the adversary is committed to excluding the non-compliant votes, even when the attack is unsuccessful, non-compliant votes are not rewarded. W_p denotes the *proposer boost*, equal to 40% of the slot committee size, that puts a temporary weight on new proposals.

10.1.3.1 Simple Attack (Section 10.4, Figs. 10.1a and 10.1b)

Suppose the leader of some slot $t+1$ is adversarial and aims to reorg the block B proposed at slot t. Before slot t starts, it informs the slot t attestors that it is the leader of slot $t+1$, and asks them to vote for B's parent block instead of B itself. The adversary commits to including in its block only the votes that *comply* with its instruction. The slot t attestors are incentivized to comply as otherwise, they do not receive the head vote reward[2]. We show a Nash equilibrium in which the attack is successful and the adversary receives the attestation inclusion reward and an additional MEV boost, given both non-colluding (called *solo*) attestors and staking pools of colluding attestors.

10.1.3.2 Strong Simple Attack (Section 10.5)

Since complying with the adversary is a weakly dominant strategy for the attestors in the simple attack, there exist other equilibria, where a coalition of players can refuse to comply with the adversary to foil the attack. Hence, in Section 10.5, we introduce a strong simple game, where an adversary controlling multiple leaders can incentivize the attestors to follow the adversarial strategy regardless of the actions of other attestors, and there is no equilibrium where the adversary fails.

[2]Head vote reward constitutes 1/4 of the total attestation reward [10], which could be the difference between making a profit vs. loss while validating new blocks.

10.1.3.3 Extended Attack (Section 10.6, Fig. 10.2)

In the extended attack, the adversary reorgs a sequence of consecutive blocks with *empty* ('transaction-less') blocks. To give insight into the attack, we describe a sketch, where some slot t is normalized to be slot 0, and the adversary is the leader of slot 3. It aims to reorg the two consecutive blocks B_{-1} and B_0 from slots -1 and 0 with a fork of empty blocks B_1 and B_2 proposed on top of block B_{-2}. Using its block B_3, the adversary can incentivize not only the slot 2 attestors but also the leaders and attestors of previous slots to follow its instructions; since the slot 2 votes affect the confirmation status of the slot 2 block, which in turn influences the rewards of the slot 1 attestors.

Formally, the adversary devises a set of instructions for the leaders and attestors of slots 1 and 2 to propose and vote for a fork of empty blocks extending B_{-2} rather than the tip B_0 of the canonical chain. Such blocks and votes that comply with the adversary's instructions are called *compliant*. The adversary stipulates the slot 1 leader to propose an empty block B_1 on top of B_{-2}, and the slot 1 attestors to either vote for B_1 if it is empty and extends B_{-2}, or to vote for B_{-1} otherwise. Now, if B_1 is compliant and gathers sufficiently many votes, the adversary demands the slot 2 leader to propose an empty block B_2, including the compliant slot 1 votes, on top of B_1. It then asks the slot 2 attestors to either vote for B_2 if it is empty, includes the compliant slot 1 votes and extends B_1, or to vote for B_1 otherwise (if B_1 is compliant). Finally, the adversary commits to including only the compliant slot 2 votes in its block B_3. Note that no rational attestor is obliged to comply with the instructions *a priori*, but as we show next, they are all *incentivized* to do so.

Due to the adversary's commitment, to receive head vote rewards, the slot 2 attestors vote for B_2 only if B_2 is compliant, *i.e.*, is empty and contains only the compliant slot 1 votes. Therefore, to ensure that B_2 receives enough votes

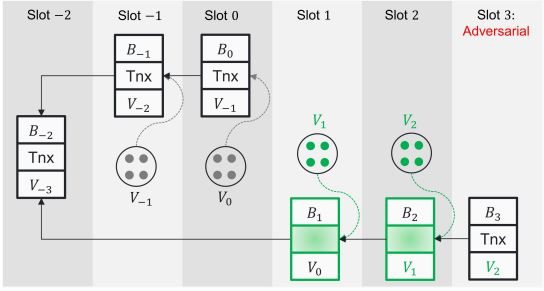


Figure 10.2: The extended attack with $p = 2$. B_i and V_i represent the block and votes for slot i, respectively. The votes and blocks in green are compliant.

and thus cannot be reorged by B_3, the slot 2 leader proposes a compliant block B_2. Since a compliant B_2 includes only the compliant slot 1 votes, to receive head vote rewards, the slot 1 attestors vote for B_1 only if B_1 is compliant, *i.e.*, is empty and extends B_{-2}. Then, to ensure that B_1 receives enough votes and thus cannot be reorged by B_2 or B_3, the slot 1 leader proposes a compliant block B_1. Consequently, all slot 1 and 2 leaders and attestors vote for a chain of empty blocks (B_1 and B_2) conflicting with B_{-1} and B_0, enabling the adversary to reorg B_{-1} and B_0. As shown in Section 10.6, the extended attack results in a subgame perfect Nash equilibrium with arbitrarily long reorgs.

10.1.3.4 Staking Pools (Section 10.7)

The simple and extended attacks assume the existence of only solo validators. However, in practice, staking pools control a large subset of validators, which can collude to mitigate the attacks, indicating a need to revisit the analysis. In Section 10.7, we illustrate Nash equilibria for both the simple and extended attacks, where the adversary succeeds even in the presence of staking pools of bounded size. However, the extended attack cannot be considered a practical threat if sufficiently many attestors coordinate their strategies. To this end, we present a new, selfish mining-inspired attack in Section 10.7, where all staking pools, regardless of their size, are incentivized to comply with the adversary under mild assumptions on the fraction of adversarial validators.

10.1.3.5 Attack Quantification (Appendix 10.A2)

Our analysis of both the consensus layer reward and the execution layer rewards shows that upon a successful reorganization of a single block, a single adversarial slot leader can achieve an additional reward of 71.1 million Gwei (170 EUR[3]), representing a 56% increase in the block reward. To better illustrate the economic impact behind reorg attacks, consider that if the largest Ethereum staking pool[4] manages to reorg one block for each of its blocks, it could collect, on average, an extra reward of almost 234,000 EUR per day.

10.1.4 A New Reward Mechanism: DAG Votes

The attacks above are due to the *monopoly* of a slot leader over which votes get included and subsequently rewarded by the canonical chain. Hence, the protocol can be reinforced against the leaders' abuse of power by *decentralizing* the reward mechanism [48]. Towards this goal, we propose a new vote reward mechanism for

[3] As of 23 October 2024, the price of 1 ETH is approximately 2400 EUR [17].
[4] As of October 23, 2024, the largest staking pool is LIDO, holding 27.8 of stake shares[18].

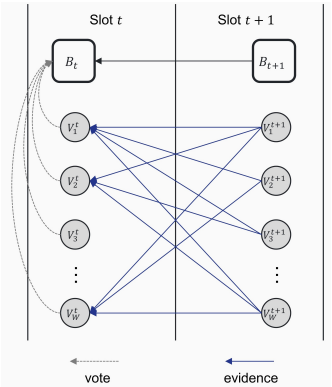


Figure 10.3: The DAG votes reward mechanism. V_i^t represents the ith vote (by the ith attestor) in slot t. Vote V_1^t has received signatures from more than half of the slot $t+1$ attestors, *i.e.*, it is timely. Vote V_3^t has not received any signature from slot $t+1$ attestors and is not timely.

LMD GHOST (Fig. 10.3, Section 10.8) called *DAG votes*, reflecting the directed acyclic graph (DAG) topology of vote connections. DAG votes replace the leader of slot $t+1$ with the slot $t+1$'s committee of attestors when determining whether slot t votes were timely. It requires each slot $t+1$ attestor to sign the slot t votes in its view. Then, an attestor is rewarded for its slot t vote if there are sufficiently many slot $t+1$ attestors that have signed the vote, *whenever* these signatures are included on-chain[5]. The adversary must now control a large fraction of the attestor committees to affect the head vote rewards. In Section 10.8.1, we prove the existence of a subgame perfect Nash equilibrium, in which LMD GHOST equipped with DAG votes remains secure under some mild assumptions.

DAG votes not only mitigate the attacks above, but also protect the honest attestors' rewards. Currently, if a slot leader does not propose any block, the attestors of the previous slot lose their head vote rewards, which is particularly discouraging for solo validators selected as attestors once per epoch. The new mechanism ensures rewards even when a minority of the attestors are offline.

10.1.4.1 DAG Votes Made Practical

Considering the number of Ethereum validators (> 1 million), the DAG votes mechanism imposes a high computational and communication overhead on

[5]The inclusion reward mechanism used by Ethereum can be extended to incentivize the inclusion of the slot $t+1$ attestors' signatures on the slot t votes.

the Ethereum network. To address this issue, in Section 10.8.2, we introduce a practical version of the mechanism that is backward-compatible with the Ethereum infrastructure and can be implemented at the cost of a slight increase in block space, computation, and communication resources. In the practical version, only a subset of the slot $t + 1$ attestors, referred to as aggregators, become responsible for signing the slot t attestations. Under synchrony, these signatures can be quickly aggregated into a single one, resulting in only a slight increase in the required block space.

In Section 10.8.3, we present an implementation of a simplified version of our DAG votes mechanism, deployed on a peer-to-peer local blockchain running across multiple Amazon Web Services instances. Our experiments show that the block space increases by merely 163 bytes due to the DAG votes mechanism, with minimal increase in the average creation and verification time for the aggregated attestations. Additionally, in Appendix 10.A6, we analyze the overhead of our scheme if implemented on the Ethereum mainnet (as opposed to a few instances as in our experiments). Our analysis shows that under the new mechanism, the block space increases by merely 14 kilobytes, which represents a 14% increase in block size. We also find that on a consumer laptop, the new mechanism imposes a computational overhead of an additional 0.69 seconds, 1.36 seconds, and 0.4 seconds for an Ethereum aggregator, block proposer, and validator, respectively, compared to the current version of Ethereum. Regarding the communication overhead of the new mechanism, each aggregator needs to send an extra pair of messages: an aggregated attestation and a signature on the aggregated attestation, resulting in only 192 extra bytes of message.

10.2 Related Work

10.2.1 Attacks on Ethereum Consensus

The LMD GHOST protocol as introduced by Buterin *et al.* in [34] is susceptible to the balancing attacks [56, 58, 59, 69], where the adversary causes two equally-sized sets of honest validators to vote for conflicting blocks indefinitely under synchrony by carefully timing the release of a few withheld adversarial votes. The protocol was subsequently changed to bestow the latest block proposals with a temporary extra weight called the proposer boost, so as to push the validators towards the latest proposal [77]. However, proposer-boosting failed to prevent a resurgence of the balancing attack due to a feature of the LMD rule that requires each validator to consider only the first slot t vote it receives from an equivocating validator. The absence of this rule in turn enabled a so-called avalanche attack that showed the incompatibility of the PoS Sybil resistance mechanism with the GHOST fork-choice rule [60]. Finally, these

attacks were addressed with another patch called equivocation discounting, removing equivocating votes for the same slot but different blocks from the validators' views [77]. A confirmation rule and the associated security proof for LMD GHOST are provided in [26].

In our work, we investigate the effect of the reward mechanism on the protocol's security *in the presence of rational validators*. Our attacks do not require the adversary to control more than a single block proposer or send equivocating blocks/votes, making the bar for the attack much lower than the earlier consensus attacks. In fact, when the attack is successful, the adversary does not even have to violate the protocol.

10.2.2 Analogous Attacks on PoW Blockchains

The attacks introduced in this paper are similar to those in PoW blockchains, such as selfish mining [42], undercutting [36] and bribery attacks [29], which aim to orphan non-adversarial blocks. In the selfish mining attack [42], the adversarial miner withholds a newly mined block and releases it later to orphan other miners' blocks. Extensive literature [51, 55, 65, 66, 75, 78] exists analyzing the impact of selfish mining on the security of longest-chain protocols (Nakamoto consensus), such as Bitcoin. In the undercutting attack [36, 44], the adversarial miner attempts to orphan the tip of the chain by incentivizing others to mine on its block during a fork race, using transaction fees as a form of bribery. In bribery attacks [29], the adversary bribes miners to temporarily control a majority of the hash power and attack the chain.

Note that there is an important difference between the incentives for orphaning (or reorging) blocks in PoW blockchains and those in Ethereum. In PoW, the primary reason orphaning blocks is profitable is that it reduces mining difficulty, thereby accelerating the process of mining blocks and collecting rewards [45, 67]. In contrast, in Ethereum, the main incentive for reorging blocks is to steal MEV and inclusion rewards from other blocks.

10.2.3 Timing Games

Our model of validators shares some similarities with the honest-but-rational validators of [61, 70]. These validators do not strictly violate the protocol rules but delay their blocks to the maximum extent possible to capture more MEV, while ensuring their timely inclusion in the canonical chain. The advantage of well-connected validators and the resulting higher profit in MEV has serious potential to hurt fairness and decentralization by incentivizing collusions for more connectivity. Like these works, our model also assumes rational, payoff-maximizing validators, and just like delaying blocks, protocol deviations by our

rational validators are not *accountably* identifiable or *slashable* (*i.e.*, punishable financially). However, unlike honest-but-rational proposers, our validators can explicitly disobey the protocol rules that are not enforceable via slashing, *e.g.*, by refusing to propose on the tip of the canonical LMD GHOST chain.

10.2.4 Game Theory and Consensus Protocols

Behavior of distributed protocols with payoff-maximizing participants has been studied for secret sharing and multi-party computation, that were shown to have Nash equilibria with desirable properties under bounded collusions [24]. Kiayias and Stouka in [49] have introduced the concept of coalition-safe equilibria to investigate the potential increase in participant payoff through protocol deviation in Bitcoin and Fruitchain [62]. In our modelling of validators, we follow the BAR (Byzantine, altruistic, rational) model of protocol participants introduced by [25], which we respectively call adversarial, honest and rational. The same work [25] provided the first safe and live asynchronous state machine replication protocol in the presence of both adversarial (Byzantine) and rational participants. For a taxonomy of the incentive mechanisms of major blockchain protocols, we refer the reader to [46].

10.3 Preliminaries

10.3.1 Model

Given any event A, let NA denote the *complement* of the event A such that $A \cap NA = \emptyset$ and $\Pr[A \cup NA] = 1$. We define notation $[p]$, $p \in \mathbb{N}$, to denote the set $\{1, \ldots, p\}$.

10.3.1.1 State machine replication and blockchains

In state machine replication consensus protocols, a set of nodes called the *validators* agree on a growing sequence of transactions (or blocks) called the *ledger* (or chain) and denoted by ch. Each validator can thus obtain the same end state upon executing these transactions in the order determined by ch. Each chain starts with a publicly known genesis block B_0. Each block except B_0 refers to its *parent* through its hash. A block B *extends* B', denoted by $B' \preceq B$, if $B = B'$ or B' lies on the path from B to B_0. All valid blocks extend B_0. For each B, the path from B to B_0 determines a unique chain ch. The chain held by a validator val at time t is denoted by $\mathsf{ch}_t^{\mathsf{val}}$.

10.3.1.2 Adversary

The adversary $\mathcal{A}$ is an efficient algorithm that controls a subset of the validators called *adversarial*. These validators can violate the consensus rules in an arbitrary fashion coordinated by $\mathcal{A}$ (Byzantine faults). The remaining validators are either honest, *i.e.*, follow the prescribed protocol, or *rational*, *i.e.*, take the action providing the largest expected payoff in any specified game. In later sections, we will first consider a model with adversarial and *solo validators*, which are rational and do not collude. Afterwards, we will analyze the interaction between the adversarial validators and colluding sets of rational validators.

10.3.1.3 Network

We consider a synchronous network, where the adversary can control the delivery time of the messages sent by the rational and honest validators up to a known delay upper bound Δ. We hereafter normalize Δ to be 1 and adopt a lock-step communication model, where messages can only be sent at discrete intervals, and a message sent at some time t (*i.e.*, wall clock time Δt) is delivered by time $t+1$, (*i.e.*, $\Delta t + \Delta$). Since our attacks do not require the adversary to delay the honest messages in arbitrary ways, we assume that the views of all validators and the adversary are the same at all 'lock-step' times. A validator is said to *broadcast* a message if its recipients include all validators.

10.3.1.4 Security

An SMR protocol is secure with latency T_{conf} if:

Safety: For any times t, t' and validators $\mathsf{val}, \mathsf{val}'$, either $\mathsf{ch}_t^{\mathsf{val}} \preceq \mathsf{ch}_{t'}^{\mathsf{val}'}$ or vice versa. For any $t' \geq t$, $\mathsf{ch}_t^{\mathsf{val}} \preceq \mathsf{ch}_{t'}^{\mathsf{val}}$.

Liveness: If a transaction tx is input to an honest validator at some time t, then there exists a finite time $t^* > t$ such that $tx \in \mathsf{ch}_{t'}^{\mathsf{val}}$ for any $t' \geq t$ and any val.

10.3.2 The LMD GHOST Protocol

The LMD GHOST protocol proceeds in *epochs* with 32 *slots*, each with duration $3\Delta = 3$. Before an epoch e, a randomness beacon called RANDAO divides the validator set of epoch e into 32 disjoint *committees* of size W assigned to 32 unique slots within the epoch [12]. It also selects a validator from each committee as the unique *leader* for its slot. A slot with an adversarial (rational,

honest) leader is called an *adversarial* (rational, honest) slot. The adversary knows which slots are adversarial before the epoch starts[6].

Validators use the LMD GHOST fork-choice rule to identify a *canonical* (beacon) chain of blocks (later, Casper FFG finalizes the blocks on this canonical chain and thus outputs the finalized chain). As we assume that all validators have the same view of the protocol, we hereafter omit the mention of views. At the beginning of a slot t, the leader L signs and proposes a new block $B = \langle e, t, \mathsf{txs} \rangle_L$ with transactions txs, that extends the canonical chain tip. After Δ time, each slot t attestor val signs and broadcasts a vote $\langle e, t, B \rangle_{\mathsf{val}}$ for the block at the tip of the canonical chain. Within Δ time, these votes are received and aggregated by a special subset of attestors, who in turn broadcast the aggregated votes to be included by the next slot's block [2]. A slot t vote is *valid* only if it is for a block with slot $t' \leq t$, and the vote was sent by a slot t attestor.

The votes sent within LMD GHOST are in fact called head votes [10] to distinguish them from the Casper FFG votes. Since we exclusively focus on LMD GHOST in this work, we will refer to head votes as simply votes unless specified otherwise.

When invoked at slot t, the fork-choice rule obtains the canonical chain iteratively starting at B_0. At each block B, it inspects the subtrees rooted at B's children. It then selects the child block, whose tree has the largest weight, ties broken by the adversary. The weight is equal to the number of unique votes by the validators for the blocks in the tree, plus a proposal boost W_{p} (currently $W_{\mathsf{p}}/W = 40\%$) if the tree contains a block proposed for the current slot t[7]. If there are multiple valid votes by the same validator within a tree, the rule ignores all but one of the votes for the latest slot, thus the name 'latest message driven (LMD)'. In future sections, we will consider slots from the same epoch, with distinct attestor committees, unless stated otherwise.

10.3.2.1 Reorgs

Suppose a block B proposed at some time t' is not in the canonical chain for the first time at some $t \geq t'$. Then, B is said to be *reorged* at time t. Reorg of a few honestly proposed blocks need not imply a liveness violation; however, reorgs of arbitrarily long block sequences would be a violation of LMD GHOST's safety and liveness (see Appendix 10.A1.3).

[6] In RANDAO, slot leaders know their slots before the epoch.
[7] The original GHOST rule counted blocks rather than votes [72]

10.3.3 The Ethereum Reward Mechanism

We review two main sources of the validator's consensus layer reward: the inclusion (proposing) and the attestation rewards. The inclusion reward is the reward that a slot leader receives for including *attestations* in its proposed block. The attestation reward is the reward that a validator receives for sending attestations. Each attestation consists of 3 different votes: the source checkpoint, the target checkpoint, and the head vote (see Section 10.3.2). An attestation is eligible to receive the attestation reward if it is included in the canonical chain and is *correct* and *timely* [10]. Here, we only elaborate on the correctness and timeliness of the head votes (hereafter called just the votes), moving the discussion of these attributes for other votes to Appendix 10.A1.1.

A slot s vote for a block is correct if that block is the last block within the canonical chain from slots $s' \in [0, s]$. If the tip of the canonical chain in the view of an attestor is extended by a block B_t at slot t, the attestor should vote for block B_t. However, if the slot block is missing, the attestor can vote for the previous block in the canonical chain, and the vote is still correct. Since Ethereum's Altair upgrade [3], a slot t vote must be included in the *subsequent* block of slot $t+1$ to be considered timely. A complete overview of the Ethereum reward mechanism is presented in Appendix 10.A1.1.

10.4 The Simple Attack in the Presence of Solo Validators

The attacks exploit Ethereum's reward mechanism to incentivize validators to act in favor of the adversary. Each attack can be viewed as a 'game' induced by the adversary that commits to a fixed set of actions and communicates this to the validators. The players are the rational validators that best respond to the actions committed by the adversary. Depending on the game, the players can include the slot leaders, attestors or both. The possible actions available to the players are as follows:

- A slot t leader can build a *single* new block for slot t, denoted by B_t, extending any block in its view at any time.
- A slot t attestor can send a *single* slot t vote for any block in its view at any time.

We assume that a rational validator never proposes two different blocks for the same slot or sends two or more votes with the same slot number for distinct blocks. This is because these violations are provably detectable and cause the slashing (burning) of the validator's stake. Although the honest actions for the slot t leader and attestors are to respectively send their block and votes at

times $3t$ and $3t+1$, the rational validators can delay their messages or withhold them if that gives them a higher payoff.

The players' payoffs come from the inclusion and attestation rewards (see Section 10.3.3). We denote by r the reward an attestor receives for submitting a single correct and timely vote and by R the reward a slot leader receives for including a single correct and timely vote in its block.

10.4.1 The Simple Game

The simple game with solo validators (Fig. 10.1a) starts at the beginning of some slot t at time $3t$ such that the leader of slot $t+1$ is adversarial. The players are the slot t attestors that respond to the following strategy by the adversary:

Definition 10.4.1 (Adversary's Strategy for the Simple Game)**.** Let B_{t-1} denote the block at the tip of the canonical chain at the end of slot $t-1$. At the start of slot t, the adversary $\mathcal{A}$

1. Generates a proof asserting its leadership in slot $t+1$.
2. Sets the instructions for the validators (called the game rule) as follows: The attestors of slot t should vote for the block B_{t-1} by the end of slot t.
3. Commits to the following action: Votes of the attestors that do not vote for B_{t-1} will not be included in the adversary's block B_A in slot $t+1$.
4. Broadcasts the leadership proof, the game rule and its commitment to all slot t attestors.

At the start of slot $t+1$, adversary $\mathcal{A}$

1. Proposes a block B_A that includes only the slot t votes for the block B_{t-1}.
2. If W_{p} or more slot t attestors have voted for B_t (block of slot t) by time $3t+3$, it proposes its block B_A on top of block B_t[8]. In this scenario (Fig. 10.1a), the attack **fails**.
3. If less than W_{p} attestors have voted for B_t by time $3t+3$, it proposes its block B_A on top of B_{t-1}. In this scenario (Fig. 10.1b), the attack **succeeds**.

The simple game ends, and the payoffs are realized at the end of slot t, at time $3t+3$. Although the rewards for the votes included in B_A are actually realized upon B_A's finalization, we assume that the behavior of the adversarial and rational validators after the game ends does not affect B_A's finalization, which happens within a short period ($\sim$ 12 minutes for Casper FFG).

[8] In this case, the compliant votes included in B_A only receive the source and target checkpoint rewards and do not receive the head vote reward. This is because the votes are for B_{t-1} and do not satisfy the correctness property, as they are not for B_A's parent B_t.

Table 10.1: Pay-off matrix of a solo slot t attestor in the simple game. Let C_{val} denote the event that the validator val votes for B_{t-1} (*i.e.*, val is compliant). Let `Succeed` denote the event that fewer than W_{p} slot t attestors vote for B_t (the attack succeeds), and `Fail` denote the event that greater than or equal to W_{p} slot t attestors vote for B_t (the attack fails).

	C_{val}	NC_{val}
`Succeed`	r	0
`Fail`	0	0

10.4.2 The Simple Game Analysis

The slot t attestors who follow (do not follow) the game rule above are called compliant (non-compliant).

Theorem 10.1. There exists a Nash equilibrium of the simple game in the presence of solo validators, where the attack succeeds.

Proof of Theorem 10.1. The pay-off matrix of a solo slot t attestor val is presented by Table 10.1. If over W_p slot t attestors vote for B_t, then B_A extends B_t and does not include any of the votes by the non-compliant attestors (`Fail`-NC_{val} corner). In this case, none of the votes by the compliant attestors included in B_A can be correct either, as they would be for the parent of B_t rather than B_t itself (`Fail`-C_{val} corner). Thus, the payoff is 0 for both actions C_{val} and NC_{val} when attack fails. In contrast, if less than W_p slot t attestors vote for B_t, B_A extends B_{t-1} and stays in the canonical chain after slot t. Therefore, the compliant attestors, whose votes within B_A would be correct and timely, receive a payoff of r (`Succeed`-C_{val} corner), whereas no votes by the non-compliant attestors are included in B_A, making their payoff 0 (`Succeed`-NC_{val} corner). Hence, voting for B_{t-1} weakly dominates voting for B_t or other blocks, and the case where solo slot t attestors follow the game rule is a Nash equilibrium. □

Note 10.1. There are many Nash equilibria, where the adversary $\mathcal{A}$ is not successful. Suppose over $W_{\mathsf{p}} + 1$ slot t attestors vote for B_t (the attack fails). Then, $\mathcal{A}$ cannot reorg B_t with B_A, and no vote for a block other than B_t can be both correct and timely. As $\mathcal{A}$ also commits to excluding votes for B_t, no solo slot t attestor receives a positive payoff for any action. Therefore, voting for B_t weakly dominates any other action for the slot t attestors, making this a Nash equilibrium.

Note 10.2. Even under an honest minority assumption (*i.e.*, $< W_{\mathsf{p}}$ honest attestors per slot), LMD GHOST is still susceptible to the simple attack, *i.e.*, there is a Nash equilibrium where the adversary succeeds; since the number

of honest validators would not be sufficient to overcome the proposer boost exploited by the adversary's block.

In Appendix 10.A2, we present a quantification of the reorganization attack to estimate the average reward an adversary gains from a successful simple game attack, as well as the cost incurred in the event of an unsuccessful attempt.

10.4.3 Credibility

So far, we have considered an adversary that commits to carrying out a certain action. Although the adversary was not modeled as a rational player, a major motivation for such attacks is financial profit: by reorging B_t, it can include a more valuable set of transactions in its block B_A, resulting in a higher profit via inclusion rewards and MEV capture. However, since a slot leader profits from including *all* correct votes due to the inclusion reward, if the adversary were financially motivated, its threat about excluding the non-compliant votes would not be considered *credible* by the rational players, unless the adversary can irrevocably commit to following upon its threat. One way the adversary can enforce credibility is by depositing collateral, larger than the maximum possible inclusion reward, in a public contract that burns the funds if the committed strategy is violated. Although this requires the adversary to deposit new collateral, this initial cost can be bypassed in Ethereum using restaking protocols.

The restaking protocols, such as Eigenlayer [16], are mechanisms that enable Ethereum stakers to leverage their staked Ether (ETH) to secure multiple services and applications simultaneously. They allow users to *restake* their existing ETH stake in various new protocols or services without having to lock up additional funds. Although restaking protocols offer enormous opportunities to Ethereum validators, they can also eliminate the initial cost of commitment attacks, a major concern for cryptoeconomic security. Indeed, using restaking protocols, an adversarial validator can reuse its ETH stake as collateral in the public contract instead of putting up new deposit. Moreover, the adversary can set the withdrawal address for its inclusion rewards to be a contract that complies with the restaking protocol, and then specify the inclusion of non-compliant votes as an additional slashing condition for burning this reward.

10.5 The Strong Simple Attack in the Presence of Solo Validators

The strong simple attack targets a slot t at time $3t$ in epoch e, such that the adversary is the leader of slot $t+1$ and is presumed to be the slot leader in at least one slot belonging to the upcoming epochs $\{e+2, e+3, \ldots\}$[9]. It is similar to the simple game, with the difference that the adversary is the leader of an additional slot $t'+1$, where $t' \in \{e+2, e+3, \ldots\}$. Slot t' should belong to an epoch $e' \geq e+2$ to ensure that the slot t' attestors are unknown at the time of the attack execution at slot t in epoch e. We denote the adversarial block of slots $t+1$ by B_A and refer to the adversarial block of slot $t'+1$, denoted by B'_A as the supporting block.

Definition 10.5.1 (Adversary's Strategy for the Strong Simple Game)**.** Let $t'+1$ denote the first adversarial slot after the start of epoch $e+2$. At the start of slot t in epoch e, the adversary $\mathcal{A}$

1. Generates a proof asserting ownership over the private keys of adversarial validators, including the adversarial leader of slot $t+1$.
2. Sets the instructions for the validators (called the game rule) as follows: The attestors of slot t should vote for the block B_{t-1} by the end of slot t.
3. Commits to the following action: The vote of a non-compliant attestor (one that does not vote for B_{t-1}) will not be included in the adversary's block B_A in slot $t+1$. Furthermore, if a non-compliant slot t attestor is among the attestors of slot t' in epoch $\geq e+2$, its vote for slot t' will also not be included in the adversary's block B'_A in slot $t'+1$.
4. Broadcasts the ownership proof, the game rule, and its commitment to all slot t attestors.

At the start of slot $t+1$, adversary $\mathcal{A}$ follows the same action as in simple game (Definition 10.4.1). At the start of slot $t'+1$, adversary $\mathcal{A}$

- Includes only the votes of those slot t' attestors who were either not among the slot t attestors or were among the compliant slot t attestors.

The strong simple game ends at the end of slot t, at time $3t+3$. However, the payoffs are realized at the end of the supporting slot t', at time $3t'+3$.

10.5.1 The Strong Simple Game Analysis

Theorem 10.2. There exists a strong Nash equilibrium of the strong simple game in the presence of solo validators, where the adversary succeeds.

[9] If the adversary's stake share is greater than ϵ, there should exist a slot after the start of epoch $e+2$ whose leader is the adversary.

Table 10.2: Expected payoff matrix of a solo slot t attestor in the strong simple game resulting from head vote rewards in both slot t and t'. Let C_{val} denote the event val is compliant. Let `Succeed` denote the event of a successful attack, and `Fail` that of a failed attack.

	C_{val}	NC_{val}
`Succeed`	$r + \frac{r}{32}$	0
`Fail`	$\frac{r}{32}$	0

Proof of Theorem 10.2. In every epoch of Ethereum consisting of 32 slots, each validator is assigned to be the attestor of exactly one slot. Therefore, a slot t attestor in epoch e can be a slot t' attestor in an epoch $\geq e+2$ with probability $\frac{1}{32}$.

The expected pay-off matrix of a solo slot t attestor val, resulting from head vote rewards in both slot t and t', is presented in Table 10.2. Vote of a slot t non-compliant attestor (regardless of the attack result) is excluded from block B_A. Also, if the non-compliant attestor is among slot t' attestors, the adversary excludes its vote from block B'_A. Therefore, the expected reward for a non-compliant attestor over 2 slots t and t' is 0 (`Fail`-NC_{val} and `Succeed`-NC_{val} corners).

Consider a compliant slot t attestor val. If fewer than W_p slot t attestors vote for B_t, the attack succeeds. In this case, block B_A includes the slot t vote of val, which is considered a correct vote. Additionally, if val is among the slot t' attestors, its slot t' vote is also included in block B'_A. Therefore, the expected reward for a compliant slot t attestor over 2 slots t and t' under a successful attack is $r + \frac{r}{32}$ (`Succeed`-C_{val} corner).

Assume that val is a compliant slot t attestor. If W_p or more slot t attestors vote for B_t, the attack fails. In this case, the slot t vote of val is considered a non-correct vote. However, if val is among the slot t' attestors, its slot t' head vote will still be included in block B'_A. Therefore, the expected reward for a compliant slot t attestor over 2 slots t and t' under a failed attack is $\frac{r}{32}$ (`Fail`-C_{val} corner). Hence, voting for B_{t-1} strongly dominates voting for B_t or other blocks, and the case where solo validators follow the game rule is a strong Nash equilibrium. □

10.6 The Extended Attack in the Presence of Solo Validators

We next describe the extended attack, where the adversary engages with the validators of multiple consecutive slots to reorg multiple blocks.

10.6.1 The Extended Game

The extended game starts at the beginning of some slot $t+p$ at time $3(t+p)$, where the leader of slot $t+2p+1$ is adversarial. We normalize $t+p=0$. Let $B_i, i \in \{-p,\ldots,0\}$, denote the block from slot i (the non-normalized slot $t+i$). Suppose at the end of slot 0, the canonical chain ends with the block sequence $B_{-p},\ldots,B_0$, each block with W votes. The game has the attestors and the leaders of the slots in $[p]$[10] as its players (excluding the adversarial leader of slot $p+1$). All players are rational solo validators. The actions available to the players are as in Section 10.4.1.

The adversary *succeeds* if the canonical chain at the end of slot p is a sequence of empty blocks extending B_{-p}, *i.e.*, if the chain consisting of the blocks $B_{-p+1},\ldots,B_0$ has been reorged by a chain of empty blocks $B_1^e,\ldots,B_p^e$ from slots $1,\ldots,p$ (Fig. 10.4). It *fails* otherwise. The game ends, and the payoffs are realized at the end of slot p.

10.6.2 The Adversarial Strategy

Before describing the adversary's strategy, we define *compliant* blocks and votes (Fig. 10.4, Def. 10.6.1). To this end, we first define the concept of *compliant* tip at slot i (see Alg. 3). Consider the leader L_i of a slot $i \in [p]$ that wishes to propose a compliant block at time $3i$. To identify the compliant tip at time $3i$, L_i iterates over all blocks B in the block tree of B_{-p} and its descendants at time $3i$ (Alg. 3, Line 3). For each block B, it envisions the hypothetical canonical chain ch_B that would be returned by the LMD-GHOST fork-choice rule at the start of slot $p+1$, if *all the attestors of slots* $i,\ldots,p$ *(in total* $(p-i+1)W$*) were to vote for* B*'s descendants (but not* B*), and* $\mathcal{A}$*'s block* B_A *were to extend the tip of the chain descended from* B (Alg. 3, Lines 4 and 5). It subsequently checks if B would be part of this canonical chain ch_B (Alg. 3, Line 6). If so, L_i records the slot number of the non-compliant block with the largest slot in B's prefix in a dictionary S indexed by blocks (Alg. 3, Line 7). After repeating this procedure for all blocks in the block tree of B_{-p}, L_i identifies the block B^* such that B^* is in ch_{B^*} and the slot number of the last *non-compliant* block in

[10] We define notation $[p]$, $p \in \mathbb{N}$, as $[p] = \{1,2,\ldots,p\}$.

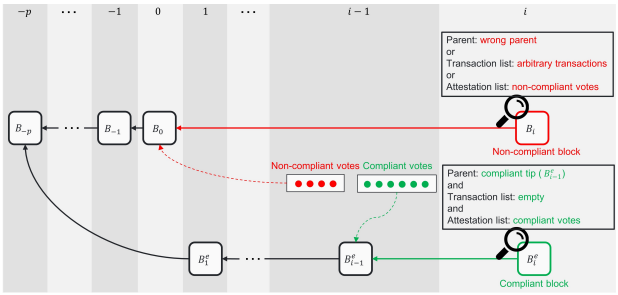


Figure 10.4: Compliant blocks and votes in the extended game. The extended game is successful if the chain of blocks $B_{-p+1}, \ldots, B_0$ gets reorged by the chain of empty blocks $B^e_1, \ldots, B^e_p$.

B^*'s prefix is smallest among all blocks considered by Alg. 3 (ties are broken deterministically). We call block B^* *the compliant tip* at slot i. The same steps are used by the slot i attestors to identify the compliant tip.

Definition 10.6.1 (Compliance)**.** A compliant slot $i \in [p]$ block is denoted by B^e_i ($B^e_0 = B_{-p}$).

- A compliant slot 1 block is *empty*, and is proposed on top of block B_{-p}.

Algorithm 3 The algorithm used by the leader and the committee members of a slot $i \in [p]$ to identify *the compliant tip*. Let $\mathcal{T}$ denote the tree of blocks that includes B_{-p} and its descendants at time $3i$. Let $B.W$ denote the weight of a block B as determined by the votes for B and its descendants. The function LMD-GHOST takes a blocktree as input and returns the canonical chain identified by the LMD-GHOST fork-choice algorithm. The function INDEX takes a chain of blocks as input and returns the slot number of the non-compliant block with the largest slot within the chain.

function PROPOSALCOMPLIANT($\mathcal{T}$)
  $S \leftarrow \{\}$
  for $B \in \mathcal{T}$ **do**
    $B.W \leftarrow B.W + (p - i + 1)W + W_\mathsf{p}$
    $\mathsf{ch}_B \leftarrow$ LMD-GHOST($\mathcal{T}$)
    if $B \in \mathsf{ch}_B$ **then**
      $S[B] \leftarrow$ INDEX(ch_B)
    end if
    $B.W \leftarrow B.W - (p - i + 1)W - W_\mathsf{p}$ ▷ Restore $\mathcal{T}$.
  end for
  $B^* \leftarrow \mathsf{argmin}_B(S[B])$
end function

- A slot i vote is compliant if it is for the compliant tip returned by Alg. 3 at time $3i + 1$.
- For slots $i = 2, \ldots, p$, a slot i block is compliant if it is *empty*, is proposed on top of the compliant tip returned by Alg. 3 at time $3i$, and contains *only* the compliant slot $i - 1$ votes for its parent.

Validators observe the protocol throughout the slots in $[p]$ and iteratively judge whether a block or vote is compliant according to the compliant tip of the slots in $[p]$ as time proceeds. Thus, they can identify the last non-compliant block at Alg. 3, Line 7.

Definition 10.6.2 (Adversary's Strategy for the Extended Game)**.** Before slot 1 starts, the adversary $\mathcal{A}$

1. Generates a proof of its leadership in slot $p + 1$,
2. Sets the game rule as follows: For each $i \in [p]$, the slot i leader should propose a compliant block, and the slot i attestors should send compliant votes.
3. Commits to excluding non-compliant slot p votes from its block B_A.
4. Broadcasts the leadership proof and the game rule to all validators of slots in $[p]$.

At the start of slot $p + 1$, $\mathcal{A}$

1. Proposes its block B_A on top of the compliant tip returned by Alg. 3 at time $3p$.
2. Includes *only* the compliant slot p votes in B_A.

10.6.3 Analysis

As the extended game lasts multiple slots and the future players observe the past actions, we show a subgame perfect Nash equilibrium (SPNE), where solo players follow the game rule.

Theorem 10.3. There exists a SPNE of the extended game in the presence of solo validators, where the adversary succeeds and violates the security of LMD GHOST.

Proof of Theorem 10.3 is provided in Appendix 10.A3. It shows that all players are incentivized to follow the adversary's game rule rather than the honest protocol at each subgame corresponding to a slot in $[p]$. Therefore, they create a chain of empty blocks $B_1^e, \ldots, B_p^e$ conflicting with the chain $B_{-p+1}, \ldots, B_0$, and thus violating LMD GHOST's safety and liveness[11].

[11]The attack considers solo validators, and as such, would work only within an epoch, *i.e.*, for $2p \leq 32$. Even in the absence of a confirmation rule for LMD GHOST, the attack violates the liveness of Casper FFG by causing arbitrarily long reorgs within an epoch.

Note 10.3. (i) There are many SPNE, where the adversary $\mathcal{A}$ is not successful, and (ii) when the number of honest validators per slot is $W_h < (W/2)$ (honest minority), LMD GHOST is still susceptible to the extended attack.

10.7 Attacks in the Presence of Staking Pools

In this section, we analyze the attacks in the presence of staking pools. We model a staking pool as a group of colluding rational validators across different slots. We call a staking pool that controls W_s attestors at each slot a W_s-staking pool. Staking pools can react differently towards the adversary's committed actions than solo validators. For instance, analyzing the simple game with solo validators, we had to consider only the reward that a solo validator received for its slot t vote. However, in the presence of staking pools, we should consider the reward in both slots $t-1$ and t as the staking pool controls attestors in both slots. Indeed, if the adversary reorgs the slot t block, the rewards of the staking pool for its slot $t-1$ votes are affected as well.

10.7.1 The Simple Game

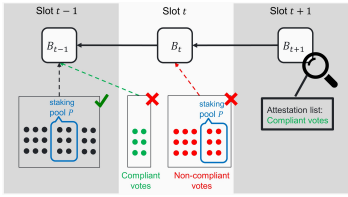
The simple game and the adversarial strategy are the same as in Section 10.4.1 except that a staking pool's payoff depends on its votes in each of the slots $t-1$ and t. The players other than the staking pool are solo attestors.

Theorem 10.4. Given a W_s-staking pool, where $W_s < W_{\mathsf{p}}$, there exists a Nash equilibrium of the simple game, where the adversary succeeds.

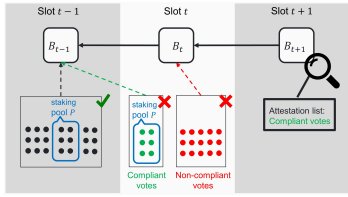
Proof of Theorem 10.4 is presented in Appendix 10.A4 and is very similar to the proof for the simple game with solo validators as $W_s < W_{\mathsf{p}}$. Fig. 10.5 illustrates the intuition behind the proof.

Note that if the staking pool controls over W_{p} attestors[12] per slot, acting non-compliantly weakly dominates any other action in the simple game, even though voting compliantly still remains weakly dominant. Therefore, we can still establish the existence of weak Nash equilibrium of the simple game in the presence of a staking pool *with any size*, where the adversary succeeds.

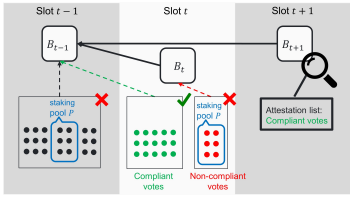
[12]As of October 23, 2024, the largest staking pool is LIDO, holding $27.8\% < W_{\mathsf{p}} = 40\%$ of the total Ethereum stake [18].



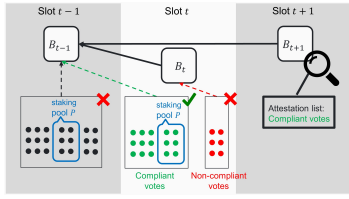
(a) `Fail`, NC_P. The payoff of staking pool P is equal to mr.



(b) `Fail`, C_P. The payoff of staking pool P is equal to mr.



(c) `Succeed`, NC_P. The payoff of staking pool P is equal to 0.



(d) `Succeed`, C_P. The payoff of staking pool P is equal to mr.

Figure 10.5: The payoff of a staking pool P with a total stake share less than W_{p} in the simple game. P controls m votes in each slot, where $m \geq 1$. r represents the reward for a single correct and timely vote. Let C_P and NC_P denote the events that the staking pool P votes and does not vote for B_{t-1} (acts compliantly and non-compliantly), respectively. Let `Succeed` denote the event that less than W_{p} slot t attestors vote for B_t (the attack succeeds), and `Fail` the event that W_{p} or more slot t attestors vote for B_t (the attack fails). The green (red) votes are compliant (non-compliant). The green tick and the red cross represent whether a vote receives or does not receive a reward, respectively. As can be seen, C_P (compliance) weakly dominates NC_P (non-compliance).

10.7.2 The Extended Game

The extended game and the adversarial strategy are the same as in Section 10.6 except that a staking pool's payoff depends on its votes in each of the slots. The players other than the staking pool are solo attestors and leaders.

Theorem 10.5. Given a W_s-staking pool, where $W_s < W_p$, there exists a SPNE of the extended game, where the adversary succeeds and violates the security of LMD GHOST.

Proof of Theorem 10.5 is in Appendix 10.A5 and has a similar outline as the proof of Theorem 10.3.

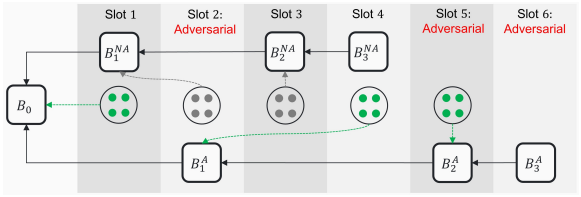


Figure 10.6: The selfish mining-inspired attack, where $p = 5$ and $N_{\mathcal{A}} = N_{\mathcal{NA}} = 3$. The upper and the lower forks are non-adversarial and adversarial, respectively. The attestors in slots 1, 4, and 5 are incentivized to vote for the adversarial fork as their subsequent slots are adversarial. At the start of slot 6, the weights of the non-adversarial and adversarial forks are $2W$ and $2W + W_{\mathsf{p}}$, respectively, resulting in the adversarial fork winning.

If the staking pool controls over W_{p} attestors, compliance with the adversary's game rule ceases to be a dominant strategy, leading to the mitigation of the attack. In the next section, we introduce the *selfish mining-inspired attack*, where even colluding validators are incentivized to comply with the adversary. In the presence of staking pools of any size, it threatens the protocol's liveness regardless of the size of the staking pool, albeit requiring a larger share of the stake.

10.7.3 Selfish Mining-Inspired Attack

In this section, we present a new, selfish mining-inspired attack that enables the adversary to do a long-range reorg attack even in the presence of large staking pools. Although the attack cannot violate safety or liveness permanently when the adversary's stake is $< 1/3$, over short periods where adversarial slots outnumber honest slots, it enables the reorg of non-adversarial blocks as in the original selfish-mining attack. Thus, it is an attack on reorg-resilience.

The *selfish mining-inspired attack* starts at the beginning of some slot t, where the number of adversarial slots in the interval $t, \ldots, t + p$ is greater than or equal to the number of its non-adversarial slots, and slot $t + p$ is adversarial. We normalize slot $t - 1 = 0$, implying that the attack occurs over the slots $[p + 1]$. The game has the attestors of those slots preceding the adversarial slots in $[p + 1]$ as its players. Let $N_{\mathcal{A}}$ and $N_{\mathcal{NA}}$ denote the number of adversarial and non-adversarial slots within the slot range $[p + 1]$. Let B_0 denote the block at the tip of the canonical LMD GHOST chain at the end of slot 0. Let s_i and $B_i^{\mathcal{A}}$, $i \in \{1, \ldots, N_{\mathcal{A}}\}$, denote the ith adversarial slot within the range $[p + 1]$ and its corresponding block, respectively. We have $N_{\mathcal{A}} \geq N_{\mathcal{NA}}$. In this attack, the adversary commits to the following strategy:

Definition 10.7.1 (Adversary's Strategy for the Selfish Mining-Inspired Game)**.** Before slot 1 starts, the adversary $\mathcal{A}$ generates a proof asserting its leadership in the adversarial slots within the range $[p+1]$. Then, for the slots $s_i - 1$, $i \in \{1, \ldots, N_{\mathcal{A}}\}$, it sets the suggested game rule $\texttt{Game}_i$ as follows: The attestors of slot $s_i - 1$ should refrain from publishing their votes at time $3(s_i - 1) + 1$ (*i.e.*, should withhold their votes). They should later vote for block $B^{\mathcal{A}}_{i-1}$ that will be published by the adversary at a point in the time range $[3p, 3(p+1)+1]$[13]. The adversary does not specify any rules for the attestors of the remaining slots.

At the start of slot p, when all the non-adversarial blocks within $[p+1]$ have been published, the adversary $\mathcal{A}$ takes the following action for $i \in \{1, \ldots, N_{\mathcal{A}}\}$:

- It publishes block $B^{\mathcal{A}}_{i-1}$ for the attestors of slot $s_i - 1$ ($B^{\mathcal{A}}_0$ is defined to be B_0).
- It collects the votes submitted by the attestors of slot $s_i - 1$ for block $B^{\mathcal{A}}_{i-1}$ and inserts them into the block $B^{\mathcal{A}}_i$ as compliant votes.

Before time $3(p+1)+1$, the adversary publishes the sequence of adversarial blocks $B^{\mathcal{A}}_i$, $i \in [N_{\mathcal{A}}]$ proposed on top of block B_0 for all the validators.

The attack *succeeds* if the canonical chain at the end of slot $p+1$ is the sequence of adversarial blocks $(B^{\mathcal{A}}_1, \ldots, B^{\mathcal{A}}_{N_{\mathcal{A}}})$ proposed on top of block B_0. The attack *fails* otherwise. The game ends, and the payoffs are realized before the voting time of slot $p+1$ starts at time $3(p+1)+1$. Similar to the selfish mining attack in longest-chain blockchains, the adversary in this attack hides the adversarial fork until the end of the attack interval. The adversarial fork publication is delayed to prevent non-adversarial slot leaders from proposing their blocks on top of it, which would allow them to avoid being reorganized. Fig. 10.6 illustrates the selfish mining-inspired attack conducted over six consecutive slots.

10.7.4 Analysis of Selfish Mining-Inspired Attack

Let the non-adversarial fork be the fork containing blocks published by the non-adversarial validators by the end of slot p, and the adversarial fork be the sequence $(B^{\mathcal{A}}_1, \ldots, B^{\mathcal{A}}_{N_{\mathcal{A}}})$. If the attestors of all slots $s_i - 1$, $i \in [N_{\mathcal{A}}]$ are incentivized to follow the adversary's game rule rather than the honest protocol, the weights of the non-adversarial fork and the adversarial fork at time $3(p+1)+1$ will be equal to $N_{\mathcal{NA}}W$ and $N_{\mathcal{A}}W + W_{\mathsf{p}}$, respectively. Then, if $N_{\mathcal{A}} \geq N_{\mathcal{NA}}$, the weight of the adversarial fork will be higher, and the attack will succeed.

[13] The time range is after all the non-adversarial blocks within $[p+1]$ have been published and before the voting time of slot $p+1$ starts.

Theorem 10.6. There exists a Nash equilibrium of the selfish mining-inspired game in the presence of staking pools, where the adversary succeeds.

Proof. Let $S_{\mathcal{A}}$ denote the set of slots whose next slot is adversarial, and let $S_{\mathcal{NA}}$ denote the set of slots whose next slot is non-adversarial. Let the staking pool P control m_s attestors in slot s. Note that the adversarial game rule implies that the staking pool should vote on the adversarial fork only in slots preceding adversarial slots, meaning that in other slots, the staking pool is free to vote on non-adversarial forks. The payoff matrix of a staking pool P in the selfish mining-inspired game is presented in Table 10.3. Following the game rule weakly dominates following the honest protocol. Therefore, a rational staking pool follows the game rule, resulting in the adversary's success. □

Note 10.4. There are Nash equilibria, where the adversary $\mathcal{A}$ is not successful. However, if in a slot interval, the number of adversarial slots is strictly greater than the number of non-adversarial ones, the selfish mining-inspired attack gives a higher payoff to the attestors in the equilibria it succeeds, as with high probability we have: $\sum_{s \in S_{\mathcal{A}}} m_s r > \sum_{s \in S_{\mathcal{NA}}} m_s r$. This implies that unlike in the extended attack, even coordination of strategies and forming stake pools might not be sufficient to disincentivize validators from following the game rules.

10.8 DAG Votes Mechanism

The DAG votes mechanism (Fig. 10.3) requires each slot $t+1$ attestor to sign all slot t head votes in its view at the beginning of slot $t+1$. A signature of a slot $t+1$ attestor on a slot t vote is called a *slot t evidence*. A slot t attestor gains an attestation reward for its vote if the vote is included in *any* slot $t' > t$ block B within the canonical chain, and the vote is *correct* and *timely* with respect to B. Correctness and timeliness are re-defined as follows:

Table 10.3: Pay-off matrix of a staking pool P in the selfish mining-inspired game. Let C_P and NC_P denote the events that the staking pool P follows the adversarial game rule (being compliant) and the honest protocol (being non-compliant), respectively. Let `Succeed` denote that the adversarial fork at time $3(p+1)+1$ outweighs the non-adversarial fork. Conversely, `Fail` denotes that the non-adversarial fork at time $3(p+1)+1$ outweighs the adversarial fork.

	C_P	NC_P
`Succeed`	$\sum_{s \in S_{\mathcal{A}}} m_s r$	0
`Fail`	$\sum_{s \in S_{\mathcal{NA}}} m_s r$	$\sum_{s \in S_{\mathcal{NA}}} m_s r$

Correctness: The slot t vote is correct if it is for the highest block from slots $\leq t$ in the prefix of B (let B_ℓ denote this block).

Timeliness: The slot t vote is timely if either B' is of slot $t+1$, or there are over $W/2$ unique slot t evidences for the slot t vote in the prefix of B' after block B_ℓ.

As the mechanism relies on slot $t+1$'s committee to infer whether a slot t vote is timely, it reduces the adversary's influence over the attestation rewards, unless it controls half of the committee. An adversary with bounded stake cannot delay the inclusion of evidences in the chain indefinitely, implying that its previous attacks fail (see Sections 10.4, 10.6 and 10.7.3). The DAG votes mechanism also allows rewarding the slot $t+1$ committee for generating slot t evidences and the leaders for including the past votes and evidences.

10.8.1 Analysis of the DAG Votes Mechanism

To show the effectiveness of DAG votes, we prove the existence of a subgame perfect Nash equilibrium in an idealized model, in which LMD GHOST equipped with DAG votes remains secure under the following assumptions:

1. A slot t vote is considered 'timely', only if there are over $W/2$ unique slot t evidences for the vote, where W is the number of attestors per slot. This assumption prevents a slot $t+1$ block B that was not proposed at the tip of the canonical LMD GHOST chain (called 'canonical chain' from now on for brevity) from rewarding 'late-released' slot t votes, when B later enters the canonical chain.
2. All rational, non-colluding (*i.e.*, solo) slot $t+1$ attestors create evidences for the slot t votes they observe by 2Δ time into slot t (*i.e.*, Δ time before the start of slot $t+1$) and broadcast these votes to all other attestors (they do not create evidences for the votes observed later). This is necessary to avoid penalizing the votes broadcast in a timely manner.
3. If a vote by an attestors is not detected as a 'timely' vote, that attestor is eventually penalized. This captures the inactivity leak functionality [64] ingrained in Ethereum consensus and is a necessary assumption for proving Theorem 10.7.
4. Rational leaders broadcast their blocks at the beginning of their respective slots[14].

[14]Timing games, where leaders delay their blocks for financial gain, are beyond the scope of this work. For more discussion, see Section 10.2.3

5. There is no proposer boost[15]. If instead the proposer boost is set to $W_p > 0$, Theorem 10.7 would require the number of solo attestors to be at least $1 + ((W + W_p)/2)$.

We note that the equilibrium strategies described below constitute a strong Nash equilibrium within each subgame of the extended game, up to collusions of size $(W/2) - 1$.

Theorem 10.7. Consider executions of LMD GHOST with DAG votes, where all attestors are rational, and at any given slot, over $1 + (W/2)$ attestors are solo attestors. Suppose that the slot leaders are rational, except the leader of some slot $t+1$, which is adversarial. Then, there exists a subgame perfect Nash equilibrium, where the protocol satisfies security (*i.e.*, safety and liveness).

Proof. We will show the existence of a subgame perfect Nash equilibrium (SPNE), where the solo attestors vote for the block at the tip of the canonical chain at the voting time of their respective slots, and the rational slot leaders create blocks at the tip of the canonical chain at all slots. In this setting, the adversarial leader cannot cause a reorg of the canonical chain, as its block will fail to gather any votes, if it is not placed at the tip of the canonical chain at slot $t + 1$.

For backward induction, suppose that rational leaders and attestors after some slot s follow the prescribed strategy above.

Lemma 10.1. For any slot s, votes of all attestors of the slots less than s are observed by the slot s attestors by the start of slot s. Moreover, all attestors send timely votes.

Proof. By assumption (1), a slot $s' < s$ vote is considered timely only if there are over $W/2$ unique slot s' evidences by the slot $s' + 1 \leq s$ attestors. Now, by assumption (3), a rational attestor incurs a loss if its vote is not considered timely. In contrast, its payoff for a timely vote is zero in the worst case. More specifically, a rational attestor gains a positive reward if its vote is both correct and timely; if the vote is timely but not correct, its payoff becomes zero. Therefore, in all slots $s' < s$ (as well as later slots), rational attestors ensure that their vote will be considered timely, *i.e.*, there will be more than $W/2$ unique slot s' evidences from the attestors of slot $s' + 1$. This implies that a solo slot $s' + 1$ attestor must have observed the slot s' vote Δ time before the start of slot $s' + 1$, which by assumption (2), is broadcast to and heard by all other attestors by the start of slot $s' + 1 \leq s$. Then, as all attestors are rational, votes of all attestors of the slots less than s are observed by the slot s attestors by the start of slot s. □

[15]Although LMD GHOST is not secure against Byzantine attestors in the absence of proposer boost, it is not necessary for Theorem 10.7 that considers rational attestors.

Subgame for slot s attestors: By Lemma 10.1, all solo slot s attestors observe all votes sent for slots $< s$ by the start of slot s (*i.e.*, all past votes are public). Therefore, when solo slot s attestors vote for the tip of the canonical chain at the voting time of slot s, they end up voting for the same block B, which is at the tip of the canonical chain according to the votes from slots $\leq s$. Now, by the assumption that later slot leaders and attestors follow the prescribed strategy, block B and its descendants gain over $1 + (W/2)$ votes at each slot $s' \in [s, t]$, thus remain in the canonical chain. We note that even if $s = t$, the adversary cannot reorg B; since (i) by the start of slot $t + 1$, all votes of the previous slots will be public by Lemma 10.1, and (ii) B will be at the tip of the canonical chain with an advantage of at least one vote at that time, implying that the solo attestors of slot $t + 1$ will not vote for the adversary's block unless it extends B.

Now, by assumption (4), block B is the latest block on the canonical chain from slots $\leq s$. Thus, the votes of the solo slot s attestors are correct (and timely by Lemma 10.1), implying that they receive a reward of $r > 0$ for their votes. In contrast, if a solo slot s attestor does not vote for B, its vote would not be correct, implying a reward of ≤ 0. Hence, there exists an equilibrium of the subgame of the slot s attestors, where all solo slot s attestors send timely votes for the canonical chain tip in their views.

Subgame for slot s leader: Suppose all attestors of the slots $s' \geq s$ and the leaders of the slots $s' > s$ follow the prescribed strategy above, and consider the payoff of the slot s leader. By Lemma 10.1, the slot s leader observes all votes sent for slots $< s$ by the start of slot s. Then, if the leader places its block B at the tip of the canonical chain, by the assumption on the slot $s' \geq s$ attestors and the slot $s' > s$ leaders, B and its descendants remain part of the canonical chain, implying a reward of $R > 0$ for the leader (see the reasoning in the paragraph above). Otherwise, the leader's reward would be 0. Therefore, there exists an equilibrium of the subgame of the slot s leader, where it proposes the slot s block at the tip of the canonical chain in its view.

Finally, by backward induction, all solo attestors vote for the block at the tip of the canonical chain, and rational slot leaders create blocks at the tip of the chain in all slots up to the slot $t + 1$. Moreover, all of these blocks remain as part of the canonical chain. This implies that the prescribed strategy is an SPNE, where the protocol satisfies safety and liveness. □

10.8.2 DAG Votes Made Practical

We now present a practical version of DAG votes and compare its performance with Ethereum's reward mechanism. Recall that an attestation includes a signature on the head, source checkpoint, and target checkpoint votes.

10.8.2.1 Current version of Ethereum

Let N denote the total number of validators[16]. Each slot has $N/32$ attestors, which are divided into 64 sub-committees, each with $\frac{N}{32*64}$ attestors. A subset of the sub-committee is selected to be the aggregators of the sub-committee[17]. These aggregators are responsible for aggregating the attestations (signatures) of their sub-committee. Ethereum uses the BLS signature for attestations due to its key-homomorphism [28]. In the ideal scenario, where all attestations of the sub-committee members are the same, these attestations can be aggregated into a single aggregated attestation (aggregate for short). Each aggregate contains an aggregated signature and an aggregation list. The ith element in the aggregation list is 1 if the signature of the ith attestor in the corresponding sub-committee is included in the aggregated signature, and 0 otherwise. In the ideal scenario, there exist 64 aggregates per slot in total. Each block can include up to 128 aggregates, where the slack between 64 and 128 can cover attestations for different blocks or prior slots[18].

10.8.2.2 Practical version of DAG votes

In our solution introduced in Section 10.8.2.2, all the attestors in each slot are responsible for generating evidences. This can increase the communication and storage overheads drastically. To address this issue, we propose a practical version of DAG votes. In this version, a subset of attestors is sampled in each slot to serve as the *aggregators* responsible for generating the evidence. In contrast to the main version, only the aggregators produce the evidences in the practical version.

Let sub-committee (t, i) represent the ith sub-committee of slot t, N^{att} denote the number of attestors per sub-committee, N^{agg} denote the number of aggregators per sub-committee. An attestation must gather over N^{limit} evidences to be considered timely. Each attestor in sub-committee $(t+1, i)$ sends an attestation at time $3(t+1)+1$. Then, the aggregator $j \in [N^{\text{agg}}]$ in sub-committee $(t+1, i)$ takes the following actions at time $3(t+1)+2$:

- Collect the individual attestations of sub-committee $(t+1, i)$ members and aggregate them into a single aggregated attestation (aggregate) denoted by $\mathsf{Ag}^{j}_{t+1,i}$.
- Generate an evidence for slot t attestations if the block of slot $t+1$ is not published or excludes some of the slot t attestations. Collect the aggregates by the aggregators of sub-committee (t, i), *i.e.*, $\mathsf{Ag}^{k}_{t,i}$ for $k \in [N^{\text{agg}}]$, select

[16] As of October 20, 2024, there are 1,073,375 active Ethereum validators [21].
[17] Average number of aggregators in each sub-committee is 16.
[18] These attestations may be eligible to get the source/target rewards.

and sign the one with the greatest number of signatures to generate the jth sub-committee (t, i) evidence denoted by $E^j_{t,i}$.

- Broadcast slot $t+1$ aggregate $\mathsf{Ag}^j_{t+1,i}$ and slot t evidence $E^j_{t,i}$.

If slot t attestations are not included in the slot $t+1$ block, any block proposed at slots $t' > t+1$ can include the slot t evidences. Each evidence is composed of 3 parts: the signature of the evidence generator, the aggregated attestation signature, and an aggregation list, specifying which validators' attestations are included in the aggregated signature.

The parameters N^{agg} and N^{limit} should be configured to ensure that each sub-committee has more than N^{limit} non-adversarial available aggregators with high probability. This ensures the adversary cannot prevent the timeliness verification of attestations and cannot prove delayed votes as timely by publishing their evidences late.

10.8.3 Implementation of the DAG Votes

Due to the difficulty of modifying existing testnets for LMD GHOST, we decided to estimate the overhead ratio of evidence creation from limited experiments. In this context, we have implemented a simplified version of our DAG votes mechanism in Python [23], utilizing BLS signatures [15] over the BLS12-381 curve. This implementation features a peer-to-peer local blockchain running across multiple Amazon Web Services (AWS) EC2 instances. One instance serves as a Bootstrapping node to initialize the blockchain and assist other nodes in discovering peers.

Our implementation includes two configurations: the Ethereum basic implementation and the DAG votes mechanism. The Ethereum basic implementation replicates the behavior of Ethereum's LMD GHOST protocol, incorporating functionalities such as block proposal, vote proposal, and aggregation. For any arbitrary slot duration of 3Δ seconds, the slot t block is proposed at time $3t\Delta$, the slot t votes are broadcast at time $(3t+1)\Delta$, and the slot t aggregations are broadcast at time $(3t+2)\Delta$.

Table 10.4: Comparison of Basic Ethereum Implementation and DAG Votes Mechanism

Metric	**Basic Ethereum**	**DAG Votes Mechanism**
Block size (excluding transaction payload)	582 bytes	745 bytes
Aggregation size	400 bytes	804 bytes
Average aggregation creation time	0.0138 s	0.0142 s
Average aggregation verification time	0.0076 s	0.0117 s

The DAG votes mechanism extends this implementation by introducing an evidence generation and verification process to support the DAG votes protocol. In this setup, aggregators not only aggregate votes from the same slot but also create evidence for votes from the previous slot. Specifically, the slot t aggregation includes the aggregated signature of slot t votes and a piece of evidence for slot $t-1$ votes. In this setup, block proposers are tasked with including the aggregated evidence in the blocks they produce.

Comparison of our Ethereum basic implementation and DAG votes mechanism: We compare the performance of the basic Ethereum implementation with the implementation extended by our DAG votes mechanism in Table 10.4 to analyze the overhead of our solution. This comparison includes four metrics: block size, aggregation size, aggregation creation time, and aggregation verification time. The comparison results are obtained by running our implementation on 5 AWS EC2 instances. Each instance is of the `t3.micro` type, featuring 2 vCPUs and 1.0 GiB of memory. At each slot, one validator is randomly selected to propose a block, while all validators submit votes and aggregations. The number of aggregators per slot can be set in our implementation; however, we set it to be equal to the number of validators to ensure all validators create evidence for the previous slot's votes. A slot t aggregation in our DAG votes mechanism, in addition to including the aggregated votes of the slot t attestors, includes a piece of evidence, which is the aggregator's signature on the aggregated votes of slot $t-1$. The inclusion of this extra piece of evidence compared to basic Ethereum increases the size of the aggregation by 404 bytes. A slot $t+1$ block proposer in our DAG votes mechanism, instead of including the aggregated slot t votes as in basic Ethereum, includes the aggregated pieces of evidence generated in slot t. As the aggregated evidence has an extra signature compared to the aggregated votes in basic Ethereum, the size of the block increases by 163 bytes in our solution. The average aggregation creation and verification times are calculated based on the time taken by each validator in each slot throughout the local blockchain runtime.

Performance analysis of practical DAG votes deployed in the Ethereum mainnet: We provide estimates for the storage, computation, and communication overheads of the practical DAG votes mechanism (introduced in Section 10.8.2.2) over the current Ethereum protocol if it were to be deployed in the current Ethereum mainnet in two scenarios: (i) the optimistic scenario, where the majority of slot $t+1$ aggregators sign the same slot t aggregate, and ii) the worst-case scenario, where the slot $t+1$ aggregators sign different slot t aggregates, in Appendices 10.A6 and 10.A7 respectively.

10.9 Conclusion

As our attacks show, the ability for adversaries to commit to conditional courses of action presents challenges for the design of protocol mechanisms. This holds more so given the availability of commitment devices within the protocol's enclosure, the devices possibly executing the protocol itself. Recent literature has uncovered such attacks on fee market mechanisms, a core function of blockchains [50], naming the deployment of a smart contract committing the author to certain actions of a "Stackelberg attack", after the classic model of leader-followers games [73]. In our attacks, we assumed that the adversary is the first to commit to any strategy and communicate a credible threat to the other parties. Indeed, the order of commitments is known to be important: players will often gain advantage from the ability to commit *first* [30, 68].

The attacks presented in this paper offer a *better* response compared to the honest protocol specifications for some protocol participants, in the presence of commitment devices. We leave for future work deciding whether these attacks constitute *best* responses for rational attackers; which equilibria exist if any, when all participants benefit from the commitment ability (*e.g.*, proposers attempting to reorg one another in the spirit of undercutting attacks [36]); and the importance of the sequence of play to the determination of outcomes.

As Ethereum is an ever-evolving protocol, in Appendix 10.A8, we analyze the impact of potential future changes in the protocol (*e.g.*, block-slot voting [4], secret leader election [11], proposer-builder separation [9] and single-slot finality [13]) on our attacks, and observe that none of these modifications can mitigate them. Similarly, in Appendix 10.A9, we present a version of the simple attack without proposer boost, lest the future changes above obviate the need for proposer boost.

10.10 Responsible Disclosure

Since our work exposes vulnerabilities in the existing system, we collaborated closely with researchers at the Ethereum Foundation, a non-profit organization supporting the Ethereum ecosystem. Upon identifying these attack vectors, we promptly contacted the Ethereum Foundation, whose researchers supported our work by providing access to datasets and other resources.

10.11 Acknowledgment

We would like to thank Toni Wahrstätter for his assistance in gathering data used to quantify the MEV reward in Ethereum.

Bibliography

[1] A simple Single Slot Finality protocol. https://ethresear.ch/t/a-simple-single-slot-finality-protocol/14920, 2023.

[2] Aggregator selection. https://eth2book.info/capella/part2/building_blocks/aggregator/, 2023.

[3] Altair. https://github.com/ethereum/consensus-specs/blob/dev/specs/altair/beacon-chain.md, 2023.

[4] [block, slot] voting. https://github.com/ethereum/consensus-specs/pull/2197, 2023.

[5] Enshrine PBS. https://ethresear.ch/t/why-enshrine-proposer-builder-separation-a-viable-path-to-epbs/15710, 2023.

[6] Ethereum average block size chart. https://etherscan.io/chart/blocksize, 2023.

[7] Ethereum roadmap. https://ethereum.org/fil/roadmap/, 2023.

[8] Proof-of-stake rewards and penalties. https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/rewards-and-penalties/#slashing, 2023.

[9] Proposer-builder seperation. https://ethereum.org/fil/roadmap/pbs/, 2023.

[10] Reward of the consensus layer. https://eth2book.info/capella/part2/incentives/rewards/, 2023.

[11] Secret leader election. https://ethereum.org/fil/roadmap/secret-leader-election/, 2023.

[12] Shuffling. https://eth2book.info/capella/part2/building_blocks/shuffling/, 2023.

[13] Single slot finality. https://ethereum.org/fil/roadmap/single-slot-finality/, 2023.

[14] [AMA] We are EF Research (Pt. 11: 10 January, 2024). https://www.reddit.com/r/ethereum/comments/191kke6/comment/kh7j3ig/, 2024.

[15] Blspy library, 2024.

[16] Eigenlayer. https://www.eigenlayer.xyz/, 2024.

[17] ETH to EUR: Live Ethereum to Euro Rate. https://coinmarketcap.com/currencies/ethereum/eth/eur/, 2024.

[18] Ethereum staking. https://dune.com/hildobby/eth2-staking, 2024. Last accessed: January, 19, 2024.

[19] Flashbots transparency dashboard. https://transparency.flashbots.net/, 2024.

[20] Mev-boost. https://github.com/flashbots/mev-boost, 2024.

[21] Open source ethereum explorer. https://beaconcha.in/, 2024.

[22] Skipped slots. https://beaconscan.com/slots-skipped, 2024.

[23] Implementation of a local blockchain with a dag votes mechanism, 2025.

[24] Ittai Abraham, Danny Dolev, Rica Gonen, and Joseph Y. Halpern. Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In *PODC*, pages 53–62. ACM, 2006.

[25] Amitanand S. Aiyer, Lorenzo Alvisi, Allen Clement, Michael Dahlin, Jean-Philippe Martin, and Carl Porth. BAR fault tolerance for cooperative services. In *SOSP*, pages 45–58. ACM, 2005.

[26] Aditya Asgaonkar, Francesco D'Amato, Roberto Saltini, Luca Zanolini, and Chenyi Zhang. A confirmation rule for the ethereum consensus protocol. *CoRR*, abs/2405.00549, 2024.

[27] Christian Badertscher, Peter Gazi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In *CCS*, pages 913–930. ACM, 2018.

[28] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptol.*, 17(4):297–319, 2004.

[29] Joseph Bonneau. Why buy when you can rent? - bribery attacks on bitcoin-style consensus. In *Financial Cryptography Workshops*, volume 9604 of *Lecture Notes in Computer Science*, pages 19–26. Springer, 2016.

[30] James W Bono and David H Wolpert. Game mining: How to make money from those about to play a game. In *Entangled Political Economy*, pages 179–211. Emerald Group Publishing Limited, 2014.

[31] Ethan Buchman, Rachid Guerraoui, Jovan Komatovic, Zarko Milosevic, Dragos-Adrian Seredinschi, and Josef Widder. Revisiting tendermint: Design tradeoffs, accountability, and practical use. In *DSN (Supplements)*, pages 11–14. IEEE, 2022.

[32] Ethan Buchman, Jae Kwon, and Zarko Milosevic. The latest gossip on BFT consensus. *CoRR*, abs/1807.04938, 2018.

[33] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *CoRR*, abs/1710.09437, 2017.

[34] Vitalik Buterin, Diego Hernandez, Thor Kamphefner, Khiem Pham, Zhi Qiao, Danny Ryan, Juhyeok Sin, Ying Wang, and Yan X. Zhang. Combining GHOST and casper. *CoRR*, abs/2003.03052, 2020.

[35] Vitalik Buterin and Alistair Stewart. Beacon chain Casper mini-spec (comments #17, #19). https://ethresear.ch/t/beacon-chain-casper-mini-spec/2760/17, 2018.

[36] Miles Carlsten, Harry A. Kalodner, S. Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *CCS*, pages 154–167. ACM, 2016.

[37] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *OSDI*, pages 173–186. USENIX Association, 1999.

[38] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges. *CoRR*, abs/1904.05234, 2019.

[39] Francesco D'Amato, Joachim Neu, Ertem Nusret Tas, and David Tse. No more attacks on proof-of-stake ethereum? *CoRR*, abs/2209.03255, 2022. In FC'24.

[40] Francesco D'Amato and Luca Zanolini. A simple single slot finality protocol for ethereum. In *ESORICS Workshops (1)*, volume 14398 of *Lecture Notes in Computer Science*, pages 376–393. Springer, 2023.

[41] Francesco D'Amato and Luca Zanolini. Recent latest message driven GHOST: balancing dynamic availability with asynchrony resilience. In *CSF*, pages 127–142. IEEE, 2024.

[42] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography*, volume 8437 of *Lecture Notes in Computer Science*, pages 436–454. Springer, 2014.

[43] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *EUROCRYPT (2)*, volume 9057 of *Lecture Notes in Computer Science*, pages 281–310. Springer, 2015.

[44] Tiantian Gong, Mohsen Minaei, Wenhai Sun, and Aniket Kate. Towards overcoming the undercutting problem. In *Financial Cryptography*, volume 13411 of *Lecture Notes in Computer Science*, pages 444–463. Springer, 2022.

[45] Cyril Grunspan and Ricardo Pérez-Marco. On profitability of selfish mining. *arXiv preprint arXiv:1805.08281*, 2018.

[46] Rong Han, Zheng Yan, Xueqin Liang, and Laurence T. Yang. How can incentive mechanisms and blockchain benefit with each other? A survey. *ACM Comput. Surv.*, 55(7):136:1–136:38, 2023.

[47] Lioba Heimbach, Lucianna Kiffer, Christof Ferreira Torres, and Roger Wattenhofer. Ethereum's proposer-builder separation: Promises and realities. In *IMC*, pages 406–420. ACM, 2023.

[48] Dimitris Karakostas, Aggelos Kiayias, and Christina Ovezik. Sok: A stratified approach to blockchain decentralization. *CoRR*, abs/2211.01291, 2022.

[49] Aggelos Kiayias and Aikaterini-Panagiota Stouka. Coalition-safe equilibria with virtual payoffs. In *AFT*, pages 71–85. ACM, 2021.

[50] Daji Landis and Nikolaj I. Schwartzbach. Stackelberg attacks on auctions and blockchain transaction fee mechanisms. In *ECAI*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, pages 2073–2080. IOS Press, 2023.

[51] Michael Mirkin, Yan Ji, Jonathan Pang, Ariah Klages-Mundt, Ittay Eyal, and Ari Juels. Bdos: Blockchain denial-of-service. In *CCS*, pages 601–619. ACM, 2020.

[52] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[53] Ryuya Nakamura. Analysis of bouncing attack on FFG. https://ethresear.ch/t/analysis-of-bouncing-attack-on-ffg/6113, 2019.

[54] Ryuya Nakamura. Prevention of bouncing attack on FFG. https://ethresear.ch/t/prevention-of-bouncing-attack-on-ffg/6114, 2019.

[55] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *EuroS&P*, pages 305–320. IEEE, 2016.

[56] Joachim Neu, Ertem Nusret Tas, and David Tse. A balancing attack on Gasper. https://ethresear.ch/t/a-balancing-attack-on-gasper-the-current-candidate-for-eth2s-beacon-chain/8079, 2020.

[57] Joachim Neu, Ertem Nusret Tas, and David Tse. Snap-and-chat protocols: System aspects. *CoRR*, abs/2010.10447, 2020.

[58] Joachim Neu, Ertem Nusret Tas, and David Tse. Attacking Gasper without adversarial network delay. https://ethresear.ch/t/attacking-gasper-without-adversarial-network-delay/10187, 2021.

[59] Joachim Neu, Ertem Nusret Tas, and David Tse. Ebb-and-Flow protocols: A resolution of the availability-finality dilemma. In *SP*, pages 446–465. IEEE, 2021.

[60] Joachim Neu, Ertem Nusret Tas, and David Tse. Two more attacks on proof-of-stake ghost/ethereum. In *ConsensusDay@CCS*, pages 43–52. ACM, 2022.

[61] Burak Öz, Benjamin Kraner, Nicoló Vallarano, Bingle Stegmann Kruger, Florian Matthes, and Claudio Juan Tessone. Time moves faster when there is nothing you anticipate: The role of time in MEV rewards. In *DeFi@CCS*, pages 1–8. ACM, 2023.

[62] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *PODC*, pages 315–324. ACM, 2017.

[63] Rafael Pass and Elaine Shi. The sleepy model of consensus. In *ASIACRYPT (2)*, volume 10625 of *Lecture Notes in Computer Science*, pages 380–409. Springer, 2017.

[64] Ulysse Pavloff, Yackolley Amoussou-Guenou, and Sara Tucci Piergiovanni. Byzantine attacks exploiting penalties in ethereum pos. In *DSN*, pages 53–65. IEEE, 2024.

[65] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *Financial Cryptography*, volume 9603 of *Lecture Notes in Computer Science*, pages 515–532. Springer, 2016.

[66] Roozbeh Sarenche, Svetla Nikova, and Bart Preneel. Deep selfish proposing in longest-chain proof-of-stake protocols. In *FC (1)*, volume 14744 of *Lecture Notes in Computer Science*, pages 24–40. Springer, 2024.

[67] Roozbeh Sarenche, Ren Zhang, Svetla Nikova, and Bart Preneel. Time-averaged analysis of selfish mining in bitcoin. *IACR Cryptol. ePrint Arch.*, page 363, 2024.

[68] Thomas C Schelling. The strategy of conflict. 1960.

[69] Caspar Schwarz-Schilling, Joachim Neu, Barnabé Monnot, Aditya Asgaonkar, Ertem Nusret Tas, and David Tse. Three attacks on proof-of-stake Ethereum. In *Financial Cryptography*, volume 13411 of *LNCS*, pages 560–576. Springer, 2022.

[70] Caspar Schwarz-Schilling, Fahad Saleh, Thomas Thiery, Jennifer Pan, Nihar Shah, and Barnabé Monnot. Time is money: Strategic timing games in proof-of-stake protocols. In *AFT*, volume 282 of *LIPIcs*, pages 30:1–30:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[71] Peiyao Sheng, Gerui Wang, Kartik Nayak, Sreeram Kannan, and Pramod Viswanath. BFT protocol forensics. In *CCS*, pages 1722–1743. ACM, 2021.

[72] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *Financial Cryptography*, volume 8975 of *Lecture Notes in Computer Science*, pages 507–527. Springer, 2015.

[73] Heinrich Von Stackelberg. *The theory of the market economy*. William Hodge, 1952.

[74] Toni Wahrstätter. mevboost.pics. https://mevboost.pics/, 2024.

[75] Taotao Wang, Soung Chang Liew, and Shengli Zhang. When blockchain meets AI: Optimal mining strategy achieved by machine learning. *Int. J. Intell. Syst.*, 36(5):2183–2207, 2021.

[76] Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan-Gueta, and Ittai Abraham. Hotstuff: BFT consensus with linearity and responsiveness. In *PODC*, pages 347–356. ACM, 2019.

[77] Luca Zanolini. Evolution of the ethereum proof-of-stake consensus protocol. https://github.com/ethereum/pos-evolution/blob/master/pos-evolution.md, 2022.

[78] Roi Bar Zur, Ameer Abu-Hanna, Ittay Eyal, and Aviv Tamar. Werlman: To tackle whale (transactions), go deep (RL). In *SP*, pages 93–110. IEEE, 2023.

Appendix

10.A1 Extended Preliminaries

10.A1.1 The Ethereum Reward Mechanism

Validator rewards in Ethereum originate from two sources: the execution layer reward and the consensus layer reward. The execution layer reward refers to the total reward of a slot leader for including transactions in his block. This can be either the sum of all priority fees included in the block or the amount that a block builder pays for buying the leader's block space in a proposer-builder-separation auction [47]. The consensus layer reward includes the incentivization reward paid out to the validators for their participation in the consensus protocol execution. In this paper, we focus on the impact of consensus layer reward on the validators' behavior. Therefore, we describe the consensus layer reward mechanism below.

According to the Altair upgrade [3] of the beacon chain, a validator's consensus layer reward is made up of three sub-rewards [10]:

Attestation reward: A validator receives this for attesting to its view of the chain. Let validator val be a member of the committee assigned to slot s. It sends an attestation consisting of 3 different votes: (i) the vote for a source checkpoint specified by Casper FFG, (ii) the vote for a target checkpoint specified by Casper FFG, and (iii) the *head vote* for the block specified by LMD GHOST (see Section 10.3.2).

Proposing reward: A slot leader receives this for proposing a block if the proposed block gets included in the beacon chain.

Sync committee reward: Every 256 epochs, a group of 512 validators is randomly selected as the members of the sync committee. They sign every beacon chain block header published during the assigned 256 epochs to help light clients track the beacon chain. Sync committee members receive a reward for every slot in which they sign a block.

The attestation rewards constitute the largest part, *i.e.*, 84.4%, of the consensus layer rewards [10]. For a validator to receive the attestation reward, the attestation should be included in the beacon chain and be correct and timely as defined below [10].

Correctness implies that the attestation should agree with the view of the block proposer that includes the attestation in its block. Let an attestation belong to slot s in epoch e. Then, formally,

The source checkpoint vote is correct if the source checkpoint vote matches the justified checkpoint of epoch e (*e.g.*, the block root of the starting slot in epoch $e-1$ if epoch $e-1$ was justified).

The target checkpoint vote is correct if the source checkpoint vote is correct and the target checkpoint vote matches the target checkpoint of epoch e, *i.e.*, the block root of the starting slot in epoch e.

The head vote is correct if the target checkpoint vote is correct and the head vote matches the head of the beacon chain at slot s, namely the block root of slot s [3].

The head vote correctness rule implies that if assuming block B_{t+1} is the finalized block of slot $t+1$ in the beacon chain, then a head vote of slot t is correct if and only if the head vote matches the parent root of block B_{t+1}.

Timeliness means that the attestation belonging to slot s should get included in the beacon chain within a pre-defined slot range after slot s. This range differs for the three votes included in the attestation. Let t_1 represent the slot of an attestation (by a validator assigned to the slot t_1 committee), and t_2 represent the slot at which the attestation gets included in the beacon chain. Then,

- Head vote is timely if $t_2 = t_1 + 1$.
- Source checkpoint vote is timely if $t_1 + 1 \leq t_2 \leq t_1 + 5$.
- Target checkpoint vote is timely if $t_1 + 1 \leq t_2 \leq t_1 + 32$ [3].

The tightest slot range belongs to the head vote timeliness: if an attestation does not get included in the next slot's block, the validator loses the head vote reward. To incentivize leaders to include attestations in their blocks, the proposing reward is made proportional to the total reward of the attestations and the total reward of the sync committee outputs included in the block.

Note that when the adversary reorgs a slot t block B_t with its slot $t+1$ block B_A, it can also include the slot $t-1$ attestations that were previously included by B_t, even by non-compliant attestors, in B_A to obtain the inclusion reward for the slot $t-1$ source and target checkpoint votes.

10.A1.2 Slashing

Ethereum imposes financial punishments for certain detectable protocol violations [8, 33]. If a validator is observed to have

- proposed two different blocks for the same slot, or

- sent two or more head votes with the same slot number for distinct blocks, or
- sent two pairs of votes for source and target checkpoints such that one pair *surrounds* the other,

then, the validator is *slashed*: it is removed from the validator set, and its stake is partially burned. Slashing rules are typically used to enforce a notion of safety called *accountable safety* [33, 57, 71]:

Definition 10.A1.1. A consensus protocol satisfies accountable safety with resilience f, if when there is a safety violation, (i) at least f adversarial validators can be identified as protocol violators with the help of a cryptographic proof, and (ii) no honest validator is identified except with negligible probability.

Examples of consensus protocols with accountable safety include Ethereum, PBFT [37], Tendermint [31, 32] and HotStuff [76]. They remain safe if there are fewer than f adversarial validators (as in traditional safety); since then it would not be possible to detect f adversarial validators. These protocols come with a set of slashing rules as in Ethereum; such that if a validator violates them, then it is identified as a protocol violator and slashed. In the following sections, we assume that the rational (as well as honest) validators never satisfy any of the slashing rules of their protocols.

10.A1.3 Reorgs

Suppose an Ethereum block B proposed at some time t' is no longer part of the canonical chain for the first time at some time $t \geq t'$. Then, we say that the block B is *reorged* at time t. If B proposed at some time t' never becomes part of the canonical chain, it is said to be reorged at time t'. Reorging of a single honestly proposed block need not imply a liveness violation. For instance, Nakamoto consensus [43, 52] is prone to reorgs as shown by the selfish mining attack [42]. Yet, in every sufficiently large period, an honest block enters and stays in the longest chain indefinitely implying liveness under an honest majority. Conversely, any attack that can reorg all blocks proposed during an arbitrarily long period T results in a liveness violation. If these blocks are reorged after the period T, the attack would also violate safety under any reasonable confirmation rule. Note that this does not imply a safety violation for the Ethereum blocks *finalized* by Casper FFG. However, the safety of LMD GHOST is necessary for the stability of the chain and liveness of Casper FFG.

Ethereum is said to satisfy reorg resilience if an honest or rational leader's block enters and stays in the canonical chain at all times. Reorg resilience implies safety and liveness of the protocol assuming that there are frequent honest or rational leaders (as is the case in RANDAO) and their blocks are not empty.

10.A2 Attack Quantification

In this section, we provide an estimate of the average reward that the adversary receives via inclusion rewards and MEV capture upon a successful attack. We also quantify the cost of an unsuccessful attack due to the exclusion of non-compliant votes.

We first analyze the attestation inclusion reward. Each attestation consists of three different votes: a head vote, a source vote, and a target vote. If there is no attack, a slot $t+1$ block collects the inclusion reward for all three votes from a slot t attestation. Given the number of active Ethereum validators $N = 1,073,375$ (as of October 20, 2024 [21]) and the reward distribution formula presented in [10], the average attestation inclusion reward for a block, in the case of no attack, is equal to 0.0446 ETH (107 EUR[19]). If, under the simple attack scenario, the slot t block B_t is reorged, the adversary's block B_A in slot $t+1$ can include the attestations from both slots $t-1$ and t. In this case, B_A can collect the inclusion reward for all three votes of a slot t attestation. However, it can only collect the inclusion reward for the source and target votes of a slot $t-1$ attestation, as the head vote from slot $t-1$ would not be considered timely if included in the slot $t+1$ block B_A. Therefore, the average attestation inclusion reward for an adversarial block under a successful simple attack, where all slot t votes are compliant, can increase to 0.0777 ETH (186.4 EUR).

Besides the attestation inclusion reward, the execution layer reward can also increase under a successful attack. Figure 10.7 depicts the average MEV boost (execution layer reward) over the period from October 2023 to September 2024 for blocks whose previous block was either reorged or missing, compared to blocks whose parent block was proposed in the previous slot (data obtained from [21, 22]). The former captures the reward of an adversarial block under a successful attack, while the latter gives an estimate for the adversarial block reward under a failed attack. As can be seen in Figure 10.7, a successful simple attack can increase the MEV reward of a block from an average value of 0.082 ETH (196 EUR) to an average value of 0.12 ETH (288 EUR). Therefore, considering both the inclusion and MEV rewards, the simple attack can increase the block reward from an average value of 0.1246 ETH to an average value of 0.1977 ETH, representing an increase of 0.0711 ETH (56%) in the block reward (170.6 EUR). Note that if a staking pool manages to reorg the slot t block, it will lose the head vote rewards of those slot $t-1$ attestors that are under the control of the staking pool. Considering the largest staking pool[20], the maximum amount of head vote rewards it loses is equal to 0.0225 ETH. Therefore, by

[19] As of 23 October 2024, the price of 1 ETH is approximately 2400 EUR [17].
[20] As of October 23, 2024, the largest staking pool is LIDO, holding 27.8 of stake shares.

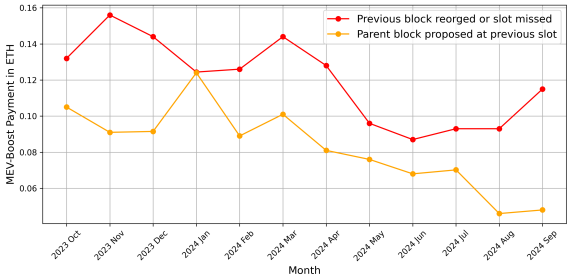


Figure 10.7: MEV reward comparison under a successful and failed simple attack.

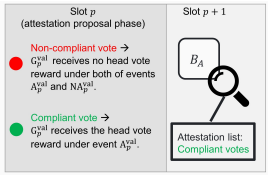
reorging a single block, a staking pool can obtain an average extra reward of at least 0.0486 ETH (116.6 EUR).

If the simple attack fails and the slot t block is not reorged, B_A in slot $t+1$ cannot include any attestations from slot $t-1$ and should also exclude the non-compliant attestations from slot t. If all slot t attestations are non-compliant, the adversary loses, on average, the attestation inclusion reward of 0.0446 ETH, but it can still get the MEV reward of a block whose parent block was proposed in time. Note that if the adversary starts the simple attack on a slot t, where both slots $t+1$ and $t+2$ are adversarial, it can include the non-compliant attestations of slot t in its slot $t+2$ block. In this case, the adversary can still induce a credible threat for slot t attestors while only losing the inclusion reward of slot t head votes, which is, on average, equal to 0.0115 ETH (27.6 EUR).

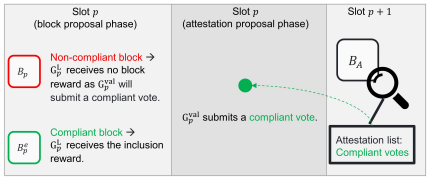
10.A3 Proof of Theorem 10.3

Proof of Theorem 10.3. Suppose all solo leaders and attestors follow the adversary's game rule and propose and vote for compliant blocks in slots $1, \ldots, p$. We show that this is a subgame perfect Nash equilibrium by backward induction. The extended game consists of subgames played by the leaders and attestors of the slots $i \in [p]$, denoted by $\mathtt{G}_i^L$ and $\mathtt{G}_i^{\mathsf{val}}$ respectively.

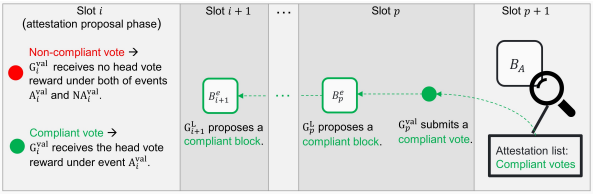
The payoff matrix of a solo slot p attestor val in $\mathtt{G}_p^{\mathsf{val}}$ is presented in Table 10.5 (Fig. 10.8a). A non-compliant vote is never included in B_A and cannot be timely, which implies a payoff of 0 (NC_p^{val} column). If val sends a compliant vote for some block B^* (C_p^{val} column) and all slot p attestors send compliant



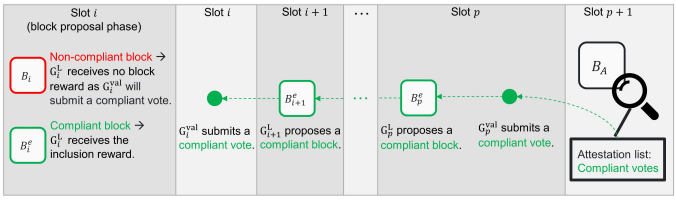
(a) Subgame $\mathtt{G}_p^{\mathsf{val}}$: C_p^{val} or NC_p^{val}?



(b) Subgame $\mathtt{G}_p^L$: C_p^L or NC_p^L?



(c) Subgame $\mathtt{G}_i^{\mathsf{val}}$: C_i^{val} or NC_i^{val}?



(d) Subgame $\mathtt{G}_i^L$: C_i^L or NC_i^L?

Figure 10.8: Subgame perfect Nash equilibrium

Table 10.5: Payoff matrix of a solo slot p validator val in the extended game. Let C_p^{val} denote the event that val sends a compliant vote. Let A_p^{val} denote the event that all slot p attestors send compliant votes.

	C_p^{val}	NC_p^{val}
A_p^{val}	r	0
NA_p^{val}	$\leq r$	0

votes[21], which must be for B^* as ties are broken deterministically (A_p^{val} row), then upon running Alg. 3, $\mathcal{A}$ obtains the same block B^* as its compliant tip and extends it with its block B_A. In this case, val gains a payoff of r (A_p^{val}-C_p^{val}

[21] In practice, the adversary can use the W_{p} boost of its block. Moreover, the blocks $\mathcal{A}$ tries to reorg will not all have the maximum W votes, implying that not all of the W slot p votes will be needed.

corner). Hence, for val, sending a compliant vote weakly dominates any other action, and it follows the game rule.

Suppose the slot p attestors send compliant votes. Then, the slot p leader L's payoff matrix in $\mathtt{G}_p^L$ is presented in Table 10.6 (Fig. 10.8b). Let R represent the inclusion reward that a slot leader receives for proposing a block in the canonical chain. If L proposes a compliant block B_p^e, all slot p attestors vote for it. Then, the compliant tip obtained by $\mathcal{A}$ includes B_p^e in its prefix, which implies a payoff of R for L (C_p^L corner). However, if L proposes a non-compliant block, the block does not gather any slot p votes. In this case, the compliant tip obtained by $\mathcal{A}$ does not include L's block in its prefix, implying a payoff of 0 for L (NC_p^L corner). Thus, proposing a compliant block B_p^e weakly dominates proposing any other block, and L follows the adversary's game rule.

Suppose all leaders and attestors of the slots $i+1, \ldots, p$ follow the adversary's game rule, *i.e.*, cast compliant votes and propose compliant blocks. Then, the payoff matrix of a solo slot i attestor val in $\mathtt{G}_i^{\mathsf{val}}$ is presented in Table 10.7 (Fig. 10.8c). Since the slot $i+1$ block is compliant, a non-compliant vote would not be included in the block of slot $i+1$, and thus cannot be timely, which implies a payoff of 0 (NC_i^{val} column).

When all leaders and attestors of the slots $i+1, \ldots, p$ follow the adversary's game rule, if val sends a compliant vote for some block B^*, (C_i^{val} row) and all slot i attestors send compliant votes (for B^*, A_i^{val} row), all of the blocks B_j^e, $j \in \{i+1, \ldots, p\}$ and B_A extend B^*, which implies a payoff of r (C_i^{val}-A_i^{val} corner). Hence, for val, sending a compliant vote weakly dominates any other action, and it follows the game rule.

Suppose all leaders of the slots $i+1, \ldots, p$, and all attestors of the slots $i, \ldots, p$, follow the adversary's game rule, *i.e.*, cast compliant votes and propose compliant blocks. Then, the slot i leader L's payoff matrix in $\mathtt{G}_i^L$ is presented in Table 10.8 (Fig. 10.8d). If L proposes a compliant block B_i^e, all slot i attestors vote for it. When all leaders and attestors of the slots $i, i+1, \ldots, p$ follow the adversary's game rule, all of the blocks B_j^e, $j \in \{i+1, \ldots, p\}$ and B_A extend B_i^e, which implies a payoff of R (C_i^L corner). However, if L proposes a non-compliant block, the block does not gather any slot p votes. In this case, the compliant tip obtained by any subsequent leader, including $\mathcal{A}$, does not include L's block in its prefix by Alg. 3, implying a payoff of 0 for L (NC_i^L corner). Thus, proposing a

Table 10.6: Payoff matrix of the slot p leader L in the extended game. Let C_p^L denote the event that L proposes a compliant block.

C_p^L	NC_p^L
R	0

Table 10.7: Payoff matrix of a solo slot i validator val in the extended game. Let C_i^{val} denote the event that val sends a compliant vote. Let A_i^{val} denote the event that all slot i attestors send compliant votes.

	C_i^{val}	NC_i^{val}
A_i^{val}	r	0
NA_i^{val}	$\leq r$	0

compliant block B_i^e weakly dominates proposing any other block, and L follows the adversary's game rule.

Therefore, there exists a subgame perfect Nash equilibrium, where all solo leaders and attestors follow the adversary's game. Then, the chain ending at the sequence of blocks $B_1^e, \ldots, B_p^e, B_A$ becomes the canonical chain after the end of slot p, and the adversary wins the game. In this case, safety and liveness are violated since the blocks $B_{-p+1}, \ldots, B_0$ are reorged. □

Note 10.5. There are many subgame perfect Nash equilibria, where the adversary $\mathcal{A}$ is not successful. Suppose the slot $i \in [p]$ leaders propose non-empty blocks B_i extending the blocks $B_0, \ldots, B_{i-1}$, and over $1 + W/2$ slot i attestors vote for B_i. Then, $\mathcal{A}$ is unsuccessful, and no slot i vote for a block other than B_i can be both correct and timely, whereas due to the adversary's commitment, no slot p vote can receive a positive payoff for any action. Thus, at each subgame, voting for B_i weakly dominates any other action for the slot i committee, and proposing a non-empty block B_i extending B_{i-1} weakly dominates any other action for the slot i leader. Therefore, the situation above is a SPNE.

Note 10.6. Under an honest minority assumption (*i.e.*, number of honest validators per slot $W_h < (W/2)$), LMD GHOST is still susceptible to the extended attack, *i.e.*, there might be a SPNE where the adversary succeeds. In the extended game, the adversary can run the game for $p' = pW/(W - 2W_h)$ blocks, gathering enough votes on a conflicting chain of p' blocks from slots $1, \ldots, p'$ to reorg the p blocks from slots $-p + 1, \ldots, 0$.

Table 10.8: Payoff matrix of the slot i leader L in the extended game. Let C_i^L denote the event that L proposes a compliant block.

C_i^L	NC_i^L
R	0

10.A4 The Simple Attack in the Presence of Staking Pools

Proof of Theorem 10.4. To analyze the simple game in the presence of staking pools, we calculate the payoff matrix of a staking pool P. Suppose the staking pool P has $m < W_p$ attestors in each slot, and recall that r represents the reward of an attestor for a correct and timely vote included in the canonical chain. Assume further that the slot t leader is not under P's control. The pay-off that a staking pool receives in the simple game is equal to the sum of the vote rewards that all its $2m$ attestors receive in both slots $t-1$ and t. The payoff matrix of staking pool P is presented in Table 10.9. Each entry in the table is calculated as the sum of two components: The left component represents the vote reward of slot $t-1$, while the right component represents the vote reward of slot t. Overall, 4 different scenarios can occur:

`Fail` – $\mathbf{NC_P}$ corner (Fig. 10.5a): W_p or more slot t attestors and staking pool P validators vote for B_t. In this case, B_t cannot be reorged by B_{t+1}, and P's votes for slot $t-1$ are included in B_t. Thus, P receives a payoff of mr for slot $t-1$. However, since the P's votes for slot t are non-compliant, the adversary $\mathcal{A}$ excludes P's votes from its slot $t+1$ block B_A. Therefore, P's reward for slot t is equal to 0.

`Fail` – $\mathbf{C_P}$ corner (Fig. 10.5b): W_p or more slot t attestors vote for B_t, but the staking pool P validators do not vote for B_t, instead voting for B_t's parent. In this case, B_t does not get reorged, and P receives a payoff of mr for slot $t-1$. As over W_p slot t attestors have voted for B_t, $\mathcal{A}$ proposes its block on top of B_t. Therefore, P does not receive any payoff for its slot t votes for B_{t-1}, as they are not correct.

`Succeed` – $\mathbf{NC_P}$ corner (Fig. 10.5c): W_p or less slot t attestors vote for B_t, but P votes for B_t. Then, the attack is successful, and B_t gets reorged. As a result, P's slot $t-1$ vote reward, *i.e.*, payoff, is equal to 0. Since P's slot t votes are non-compliant, its payoff for slot t is also equal to 0.

`Succeed` – $\mathbf{C_P}$ corner (Fig. 10.5d): W_p or less slot t attestors vote for B_t, and P does not vote for B_t. Then, the attack is successful, and $\mathcal{A}$ reorgs B_t with its block. Since block B_t gets reorged, P's payoff for slot $t-1$ is equal to 0, but it receives a payoff of mr for its compliant slot t votes.

As can be seen in Table 10.9, similar to the solo validator analysis, voting for the parent block B_{t-1} is a weakly dominant strategy for a staking pool. This indicates that if rational validators constitute over $W - W_\mathsf{p}$ of the committees, the simple attack would be successful even in the presence of staking pools. □

Table 10.9: Payoff matrix of a staking pool P in the simple game. Let C_P denote the event that the staking pool P's validators vote for B_{t-1} (*i.e.*, P is compliant). Let `Succeed` denote the event that less than W_{p} slot t attestors vote for B_t (attack succeeds), and let `Fail` denote the event that W_{p} or more slot t attestors vote for B_t (attack fails).

	C_P	NC_P
`Succeed`	$0 + mr$	$0 + 0$
`Fail`	$mr + 0$	$mr + 0$

10.A5 The Extended Attack in the Presence of Staking Pools

Proof of Theorem 10.5. Consider a staking pool P that controls $m < W_p$ attestors in each slot, and recall that r represents the reward of an attestor for a correct and timely vote included in the canonical chain. Suppose all slot $1, \ldots, p$ leaders propose compliant blocks and all slot $1, \ldots, p$ attestors vote for compliant blocks, resulting in the reorg of the blocks from slots $-p+1, \ldots, 0$. We show that this is a subgame perfect Nash equilibrium by backward induction.

Consider a branch of actions taken by the players, in which slot $1, \ldots, p$ leaders propose compliant blocks and all slot $1, \ldots, p$ attestors except those in P vote for compliant blocks (those in P might or might not have voted for compliant blocks). Let B_p^e denote the compliant block voted by the slot p attestors. Then upon running Alg. 3, $\mathcal{A}$ obtains the same block B_p^e as its compliant tip (as $W_s < W_p$) and extends it with its block B_A. Now, if P sends compliant votes, which must be for B_p^e as ties are broken deterministically, then its votes are included by B_A, and it achieves a total payoff of mr for slot p. If P does not send compliant votes, its total payoff for slot p becomes 0 as B_A would not include its votes. Hence, P also sends compliant votes for block B_p^e.

Since slot p attestors send compliant votes, the slot p leader L's block becomes part of the canonical chain only if it proposes a compliant block B_p^e, in which case it attains a reward of R. Otherwise, its reward is 0. Thus, it proposes a compliant block, *i.e.*, B_p^e.

Next, consider a branch of actions taken by the players, in which slot $1, \ldots, i$ leaders propose compliant blocks and all slot $1, \ldots, i$ attestors except those in P vote for compliant blocks (those in P might or might not have voted for compliant blocks). In this case, let B_i^e denote the compliant block voted upon by the slot i attestors. Then, the canonical chain at the start of slot $p+1$ contains B_i^e and all the compliant blocks extending B_i^e by backward induction. Now, if P sends compliant votes, which must be for B_i^e as ties are broken deterministically,

then the votes are included by the compliant block B^e_{i+1} extending B^e_i in the canonical chain, and P achieves a total payoff of $mr(p-i)$ for slots $i, \ldots, p$ by backward induction. If P does not send compliant votes, its total payoff for slots $i, \ldots, p$ becomes $mr(p-i-1)$ as B^e_{i+1} would not include the votes. Hence, P also sends compliant votes for block B^e_i.

Since slot i attestors send compliant votes, the slot i leader L's block becomes part of the canonical chain only if it proposes a compliant block B^e_i, in which case it attains a reward of R. Otherwise, its reward is 0. Thus, it proposes a compliant block, *i.e.*, B^e_i. This concludes the proof. □

Note 10.7. There are SPNE, similar to those in Note 10.3, where the adversary $\mathcal{A}$ is not successful.

10.A6 Performance Comparison: The Optimistic Scenario

In this section, we estimate the storage, computation, and communication overheads of our solution and compare them with the overheads of the current Ethereum version in an *optimistic scenario*, where the majority of slot $t+1$ aggregators sign the same slot t aggregate.

Block space: As of October 23, 2024, each block of Ethereum contains 128 aggregated attestation signatures as well as 128 aggregation lists, where the size of each signature and each aggregation list is equal to 96 bytes and $N^{\text{att}} = 524$ bits, respectively. Therefore, the total size of aggregated signatures and aggregation lists (aggregates) in a single block is 20.672 kilobytes. The average Ethereum block size from January 1, 2024, to October 23, 2024, is approximately 101.5 kilobytes [6].

Under the optimistic scenario, each block includes 128 aggregated evidences for an earlier slot (one per sub-committee in the earlier slot). Each piece of aggregated evidence comprises 4 parts: the aggregated evidence signature (96 bytes), the evidence aggregation list (N^{agg} bits), the aggregated attestation signature (96 bytes), and the attestation aggregation list (N^{att} bits). Assuming that $N^{\text{att}} = 524$, the total size of aggregated evidences in each block is equal to $32.96 + 0.016 * N^{\text{agg}}$ kilobytes. Suppose the parameters N^{agg} and N^{limit} are set such that the probability that the number of non-adversarial aggregators in a sub-committee becomes less than or equal to N^{limit} is less than 10^{-4}.

- Assuming the adversarial stake share is less than or equal to 0.1, N^{agg} and N^{limit} can be set to 16 and 8, respectively. In this case, the total size of evidences in a single block would be equal to 33.216 kilobytes. As these evidences replace the 20.672 kilobyte aggregates in the current Ethereum

version, the block size increases by 12.544 kilobytes. This results in a 12.3% increase in the block size.
- Assuming the adversarial stake share is less than or equal to 1/3, N^{agg} and N^{limit} can be set to 128 and 64, respectively. In this case, the total size of evidences per block would be equal to 35.008 kilobytes, which is 14.336 kilobytes higher than the current size of block aggregates. This results in an 14.1% increase in the block size.

Computational overhead: We assume $N^{\text{att}} = 524$ and $N^{\text{agg}} = 16$. Let C^{add}, C^{mul}, and C^{pair} denote the computational cost of an addition, a scalar multiplication, and a pairing operation over the elliptic curve, respectively. *Aggregators:* In the current Ethereum version, an aggregator must first verify the N^{att} single attestation signatures of the corresponding sub-committee, where each signature verification requires 2 pairing operations. Then, the aggregator has to perform $N^{\text{att}} - 1$ additions over the elliptic curve to aggregate attestations and 1 scalar multiplication over the elliptic curve to sign the aggregated attestation. Therefore, its computational cost becomes $2N^{\text{att}}C^{\text{pair}} + (N^{\text{att}} - 1)C^{\text{add}} + C^{\text{mul}}$. In the practical version of our solution, in addition to these operations, a slot t aggregator must verify and sign 1 aggregated attestation (the one that includes the greatest number of attestations) from slot $t - 1$ to generate an evidence. The aggregated attestation can be verified by checking its aggregated signature that contains the signatures of the attestors in the corresponding sub-committee from slot $t - 1$. To check this signature combining N^{att} individual signatures, a verifier needs to perform $N^{\text{att}} - 1$ additions over the elliptic curve to calculate the aggregated public key of the participating attestors, along with 2 pairing operations. Thus, an aggregator's computational cost in our practical version becomes $2(N^{\text{att}}+1)C^{\text{pair}}+2(N^{\text{att}}-1)C^{\text{add}}+2C^{\text{mul}}$. Based on our implementation results[22], this computation takes 0.699 seconds.

Block proposers: In our practical version, each block proposer needs to verify and aggregate N^{agg} evidences per each sub-committee to generate an aggregated evidence, in addition to standard operations of the current Ethereum. The computational costs of verification and aggregation of evidences per sub-committee are equal to $2N^{\text{agg}}C^{\text{pair}}$ and $(N^{\text{agg}} - 1)C^{\text{add}}$, respectively. As the number of sub-committees per slot is 64, a block proposer in total incurs an additional computational cost of $128N^{\text{agg}}C^{\text{pair}} + 64(N^{\text{agg}} - 1)C^{\text{add}}$. Based on our implementation results, this additional computation takes 1.36 seconds.

Verifiers: To verify the consensus layer-related data of a block in the current Ethereum version, a verifier must check 64 aggregated signatures, each belonging to one of the sub-committees. This results in a total computational cost of $64(N^{\text{att}} - 1)C^{\text{add}} + 128C^{\text{pair}}$. In our practical version, in addition to this

[22]These results were achieved through implementation on a laptop equipped with an Intel Core i7 CPU operating at a processor base frequency of 2.70GHz.

verification, a verifier must verify 1 aggregated evidence per each sub-committee, where each aggregated evidence contains N^{agg} evidences. All the evidences of a sub-committe are created over the same aggregated attestation that includes N^{att} attestations. This implies that the evidence verification cost per sub-committee is equal to $(N^{\text{att}} + N^{\text{aagg}} - 2)C^{\text{add}} + 4C^{\text{pair}}$. Therefore, and a verifier incurs the computational cost of $64(2N^{\text{att}} + N^{\text{agg}} - 3)C^{\text{add}} + 384C^{\text{pair}}$ in the practical version. Based on our implementation, this computation takes 0.402 seconds.

Communication overhead: Under the optimistic scenario, each aggregator needs to send an additional pair of messages: the aggregated attestation (96 bytes) from the previous slot and its signature on that attestation (96 bytes). This implies that each aggregator has to send 192 extra bytes compared to the current Ethereum version.

10.A7 Performance Comparison: The Worst Case Scenario

In this section, we estimate the storage, computation, and communication overheads of our solution and compare them with the overheads of the current Ethereum version in the *worst-case scenario*, which happens if the slot $t+1$ block is missing or has excluded the attestations, and the slot $t+1$ aggregators sign different aggregates. Processing these diverse evidences can impose additional overheads on the blockchain network. In the following, we present the analysis of the overheads incurred by our solution in the worst-case scenario:

Block space: In the worst-case scenario, each block includes $128 * N^{\text{agg}}$ individual evidences (N^{agg} evidences per each sub-committee). Each piece of individual evidence comprises 3 parts: the evidence signature (96 bytes), the aggregated attestation signature (96 bytes), and the attestation aggregation list (N^{att} bits). Assuming that $N^{\text{att}} = 524$, the total size of evidences in each block is equal to $32.96 * N^{\text{agg}}$ kilobytes. Suppose the parameters N^{agg} and N^{limit} are set such that the probability that the number of non-adversarial aggregators in a sub-committee becomes less than or equal to N^{limit} is less than 10^{-4}.

- Assuming the adversarial stake share is less than or equal to 0.1, N^{agg} and N^{limit} can be set to 16 and 8, respectively. In this case, the total size of evidences in a single block would be equal to 527.36 kilobytes.
- Assuming the adversarial stake share is less than or equal to 1/3, N^{agg} and N^{limit} can be set to 128 and 64, respectively. In this case, the total size of evidences in a single block would be equal to 4.218 megabytes.

Computational overhead: We assess the computational overhead under the worst-case scenario from the perspective of the following parties:

- Aggregators: It is similar to the optimistic scenario.
- Block proposers: In the worst-case scenario, each block proposer needs to verify N^{agg} evidences per each sub-committee to generate an aggregated evidence, in addition to standard operations in the current Ethereum version. The computational cost of evidence verification per sub-committee is equal to $2N^{\text{agg}}C^{\text{pair}}$. As the number of sub-committees per slot is equal to 64, a block proposer in total incurs an additional computational cost of $128N^{\text{agg}}C^{\text{pair}}$.
- Verifiers: Under the worst-case scenario, in addition to the existing verification process in the current Ethereum version, a verifier needs to verify N^{agg} single evidences per each sub-committee, where each single evidence is created over an aggregated attestation that includes N^{att} attestations. This implies that the evidence verification cost per sub-committee is equal to $(N^{\text{att}} - 1)N^{\text{agg}}C^{\text{add}} + 4N^{\text{agg}}C^{\text{pair}}$. Therefore, a verifier incurs the computational cost of $64(N^{\text{att}} - 1)(N^{\text{agg}} + 1)C^{\text{add}} + 64(2 + 4N^{\text{agg}})C^{\text{pair}}$ in the worst-case scenario.

Communication overhead is the same as the optimistic case.

10.A8 Impact of Future Ethereum Changes

In this section, we analyze the impact of potential future changes in the Ethereum protocol [7] on our attacks. None of these modifications can mitigate the attack. The attack can even become more destructive with certain changes, such as single-slot finality.

10.A8.1 [block, slot]-voting

In the current Ethereum, if no block is proposed during a slot, the slot's assigned committee votes for the previous slot's block as the head block. In the [block, slot]-voting mechanism [4], the attestors of a slot with no block vote on an *empty slot*. The [block, slot]-voting might be adopted by the Ethereum protocol in the future as it facilitates the implementation of an in-protocol proposer-builder-separation mechanism.

A slightly modified version of our attack is still applicable to the protocol with the [block, slot]-voting mechanism. For instance, in the simple game of Section 10.4.1, the game rule only needs to specify "the attestors of slot t to vote for the empty slot" rather than the previous block. Similarly, the extended game would stipulate the attestors to vote for empty slots during slots in $[p]$.

10.A8.2 Secret Leader Election (SLE)

Currently, the list of upcoming block proposers in Ethereum is public, meaning that a validator can predict future slot leaders. SLE [11] can ensure that only the selected slot leader knows its role as a block proposer in one of the upcoming slots. SLE can improve the resistance of Ethereum against the denial-of-service (DOS) attack.

SLE cannot mitigate our attack as the attacker only needs to be aware of the future time slots in which they *themselves* are selected as the slot leader, which is feasible even with the adoption of SLE. Moreover, the attack could still be modified to work under protocol changes where leaders themselves do not know their slots until the slot starts. For instance, the adversary could deploy a contract that commits to rewarding a slot t attestor, whenever the attestor can prove that the adversary is the leader of some slot $t + 1$ and that the attestor satisfies the compliance conditions with its slot t votes. The contract can also be permissionlessly triggered to slash the adversary if it happens to exclude compliant votes.

10.A8.3 In-Protocol Proposer–Builder Separation (ePBS)

Currently, Ethereum proposers (slot leaders) are responsible for creating a bundle of transactions and including them in the proposed block. They may decide to outsource the responsibility of bundle building to builders operating outside of the Ethereum protocol [9]. In this context, ePBS is a prominent direction (among many [14]) considered for the future of MEV allocation. It splits the task of building a transaction bundle and proposing a block between two validators: a block builder and a block proposer. Block builders become responsible for creating transaction bundles and offering them to block proposers. Block proposers become responsible only for proposing and sending out the blocks to peers on the network without knowing the content of the included transaction bundle. ePBS can offer a fairer distribution of MEV opportunities [5].

Our attack can also threaten the ePBS version of Ethereum. A malicious slot t leader can incentivize the attestors of slot $t - 1$ to vote on an unrevealed transaction bundle. By doing so, the victim block builder of slot $t - 1$, who has published his transaction bundle honestly, may get penalized, and the included MEV in the victim transaction bundle can be stolen.

10.A8.4 Single-Slot Finality

In present-day Ethereum, which uses Casper FFG as finality gadget, a block has to wait for 2 epochs to get finalized. The concept of single-slot finality [13]

aims to achieve the goal of finalizing blocks within the same slot they were proposed. It can thus significantly reduce the transaction confirmation time.

The security of protocols with single-slot finality [40] can be proven based on the honest majority assumption for online validators [1]. In our attacks, the adversary need not control any majority. Thus, the attacks would succeed in creating equilibria, where liveness can be violated for arbitrarily long slot sequences, on Ethereum with single-slot finality as long as the majority of the validators are profit-maximizing (rational) agents. Even though the attacks would not be able to reorg a long sequence of finalized blocks (recall that in the case of LMD GHOST, a long sequence could be reorged before the Casper FFG finalizations catch up, see Section 10.6), if the adversary is only interested in violating liveness but not safety, it can induce subgame perfect equilibria, where a long sequence of empty blocks are finalized.

10.A9 Simple Attack without Proposer Boost

The simple attack introduced in Section 10.4.1 benefits from a high proposer boost. However, even reducing the proposer boost to 0 cannot completely mitigate this attack. In this section, we introduce a modified version of the simple attack that works under the assumption of $W_{\mathsf{p}} = 0$.

Assume $W_{\mathsf{p}} = 0$. The attack applies to some slot t where slots $t-1$ and $t+1$ are adversarial. Let B_0 denote the latest block within the canonical chain before the start of slot $t-1$. In this attack, the adversary does not publish block $B^{\mathcal{A}}_{t-1}$ at slot $t-1$ and delays publishing it to the next slot. At the beginning of slot t, when the leader of slot t proposes block B_t on top of block B_0, the attacker simultaneously publishes its block $B^{\mathcal{A}}_{t-1}$ proposed on top of B_0. The adversary sets the game rule for the attestors of slot t as follows: "The attestors of slot t should vote for the block $B^{\mathcal{A}}_{t-1}$. Otherwise, their votes will not be included in the block of slot $t+1$". At the start of slot $t+1$, adversary $\mathcal{A}$ takes the following action:

- If over half of slot t attestors vote for B_t, it proposes its block $B^{\mathcal{A}}_{t+1}$ on top of block B_t and refrains from including the non-compliant votes. In this scenario, $\mathcal{A}$ is unsuccessful.
- If over half of slot t attestors vote for $B^{\mathcal{A}}_{t-1}$, it proposes its block $B^{\mathcal{A}}_{t+1}$ on top of block $B^{\mathcal{A}}_{t-1}$ and includes the compliant votes. Then, $\mathcal{A}$ is successful.

Using the same analysis as presented in Section 10.4.2, one can show as long as more than half of slot t attestors are rational, the adversary can succeed in the simple attack without proposer boost. If $W_{\mathsf{p}} > 0$, this attack can succeed even with fewer than 50% rational attestors.

Statement on the Use of Generative AI

I did not use generative AI assistance tools during the research/writing process of my thesis, except for mere language assistance. The text/code/images in this thesis are my own, and generative AI has only been used in accordance with the KU Leuven guidelines, and appropriate references have been added. I have reviewed and edited the content as needed and I take full responsibility for the content of the thesis. This sentence is obligatory.

Curriculum vitae

Roozbeh Sarenche received his B.Sc. in Electrical Engineering from the University of Tehran, Iran, in 2016, and his M.Sc. in Secure Communications and Cryptography from Sharif University of Technology, Iran, in 2019. Since July 2020, he has been pursuing a Ph.D. in Blockchain Security at the COSIC research group, KU Leuven, Belgium, under the supervision of Bart Preneel. His research focuses on the security analysis of blockchain consensus protocols and incentive mechanisms, as well as the application of cryptographic tools in blockchain systems.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
Kasteelpark Arenberg 10, bus 2452
B-3001 Leuven
roozbeh.sarenche@esat.kuleuven.be
http://www.esat.kuleuven.be/cosic

