KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Design of Scalable and Secure Decentralized Applications

Akash Madhusudan

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

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

June 2024

Design of Scalable and Secure Decentralized Applications

Akash MADHUSUDAN

Examination committee:
Prof. dr. ir. Yves Willems, chair
Prof. dr. ir. Bart Preneel, supervisor
Dr. Svetla Petkova-Nikova, supervisor
Prof. dr. ir. Frederik Vercauteren
Prof. dr. ir. Wouter Joosen
Dr. Aysajan Abidin
Prof. dr. ir. Bert Lagaisse
Prof. dr. ir. Geert Deconinck
(ELECTA, KU Leuven)
Dr. Mustafa A. Mustafa
(University of Manchester)

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

June 2024

© 2024 KU Leuven – Faculty of Engineering Science
Uitgegeven in eigen beheer, Akash Madhusudan, Kasteelpark Arenberg 10, bus 2452, B-3001 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.

In memory of my father.
The time we spent together may be short,
but his light will guide me for the rest of my life.

Preface

Growing up in India, I had aspirations that constantly changed. For the longest time, I wanted to enroll in the Indian Air Force; however, personal circumstances drove me away from this dream. This was followed by a new dream: to become an architect, but I guess by now, everyone reading this thesis knows that I never fulfilled that dream either.

Like most Indian students in 2011, I decided (well, maybe *decided* is not the right word) that I wanted to become a software engineer. Hence, I did what was required; finished my B. Tech in computer science and engineering, and got a software engineering job. Obviously, I felt that I had accomplished what I set out to do, and apparently, I was good at it too! I somehow landed a special project, which enabled me to travel to the exotic land of Belgium, where I *finally* found what I loved.

Long story short, I found the passion of *research*. I can now confidently say that, even if I am not flying the fastest jet fighters boasted by the Indian Air Force, or designing the next Taj Mahal, this has been the most rewarding experience of my life so far.

In addition to Belgium as a whole, I would like to particulary thank many people I met here, during my PhD or before it, for making this moment possible. You may not realize it, but your role in my life has helped me to reach this moment.

I would first like to express my utmost gratitude towards my supervisors, Prof. Dr. ir. Bart Preneel, and Dr. Svetla Nikova. Through your support in both *taking a chance on me* and guidance during the ups and downs of a typical PhD journey, I learned to become a more thorough researcher who pursued all questions he deemed relevant without ever being worried about the destination. An additional thanks to Svetla, who experienced my imposter syndrome first-hand, and always showed how much she believes in me. I would also like to extend my thanks to the jury for reading, evaluating, and providing valuable feedback, which greatly improved the quality of this PhD dissertation. An additional thanks to Dr. Mustafa A. Mustafa, who rallied me from the very beginning of this PhD and also at its end by being on my examination committee.

Pursuing a PhD can be a lonely journey, where each researcher shall find what they are most curious about. While the latter is true, I believe the former greatly depends on the people you are surrounded by.

I met truly wonderful people on this journey, starting with the Barracks crew. Thank you Barracks for making my PhD so incredibly fun, both while at work and especially on Friday evenings. The fact that I met a group of people equally spontaneous and crazy as myself made me fonder of COSIC, and my initial years of being a noob researcher. I am still sad I was not a part of the famous Friday crew picture at de Metafoor. Victor and Danilo, thank you for being my older brothers and helping me out through the worst of times. COVID would have been extremely depressing without you both, I passed the days looking forward to the CoD and cooking nights. Thomas, Robi, for being the best office mates. Sitting next to brilliant people every day made me stressed, but that is exactly what I needed. Péla, thank you for being there for me when I came to your office frustrated, clueless, and out of options. There are no words with which I can express your importance during my PhD, and I am sure this applies to everyone working at COSIC.

Matteo, Arno, Kevin, Cansu, Nina, and Jeroen, I am grateful to have met you during my Masters. I am sure you realize your importance in my life, and if you do not, we need to have a chat. Ishaan, I have always been in awe of how our paths crossed again, especially given the fact that the last time we hung out before Leuven was when we were 13. Thank you for always thinking of me when it came to adventure, food, gaming, and movies. The events you invited me to always gave me a break from the constant stress I learned to live with during these five years. Maite, Steffen, and Luca, you guys collectively added to the memories (yes, we do not clearly remember most of it) of several night outs in Oude Markt, Parkboostraat, and also internationally. Pakhi, I am glad to have met you through Ishaan, and cherish all our discussions where we shared stories about our imposter syndrome. You gave me confidence in the moments when I felt that I would fail, especially before my private defence. Arnout, it surprises me how similar we are. In the short time we know each other, we have made memories for years to come. Thank you for always showing up for the important occasions. Ridhima, for directly asking me about when I will finish my PhD, which always pushed me to do it; Roshni, for doing exactly what Ridhima did, but more subtly (sharing PhD memes). On a side note, I have to admit that all the Parhez's actually worked.

In November 2022, I decided that it was time I applied and further nourished the knowledge I gained in 4 years of deep academic research elsewhere. Tomer, I still do not understand the basis of your unwavering belief in me, but it has been driving me to become a better person. Thank you for your blind faith in me, and for making me a part of 3MI Labs, an organization that takes my scientific curiosity to the extreme daily. Kathy, Cyprien, Erik, Amit, and Sundas, thank you for collectively pushing me to this moment, and taking over most of my responsibilities when I needed more time to focus on finalizing my PhD. On a side note, Casomorphins really are a thing.

Last but not (at all) least, I dedicate this work to my biggest believers. Joke, you probably saw the worst side of me, and yet you persisted to somehow still find me likable. I cannot express how much I appreciate your presence in my life and all the small things you do to keep me going. Akanksha, thank you for always giving me strength. You are the strongest person I know, and I am glad to have grown up in your shadow. Your caliber, drive, kindness, and hunger for knowledge are things I aspire to match one day. Jared, thank you for extending our family and for supporting my sister during her worst and her best. Not a day passes when I do not aspire to be as funny and caring as you are. Noah, Archer, thank you for making me smile every day. You are the most important people in my life, and I promise you that Akash Mama will always be there for both of you.

Mom, I do not think I can ever repay all your sacrifices to ensure that your children got the *best* always and every day. You are not only my nourishing and kind mother, but you are also my strong and protective father. The reason that I am what I am today is because you never let me go astray. Thank you for teaching me the value of honesty, hard work, and always leading by example. Jai Guruji.

Akash Madhusudan
Leuven, June 2024

Abstract

Nowadays, one cannot imagine life without the Internet: we use it to stay connected with our social circle, find our dream job, express our opinions, and watch our favorite documentaries; we live our second, more efficient, and connected life online. Given the years of innovation the Internet has undergone, more use cases have emerged, all with one common architectural approach, i.e., a centralized architecture. Centralization of services has led to a seamless user experience, resulting in the massive growth of Internet giants such as Meta, Google, etc.

The financial engine of personalized advertisements drives the growth of the Internet. Living our second life on the Internet has also made us targets of customized advertisements, facilitated by the large-scale data mining of our every action. If such advertisements are not bad enough, data breaches that reveal sensitive information about billions of people yearly are another shortcoming of this centralized architecture.

These issues have led to a resounding call for decentralization, with blockchain technology at this movement's forefront. The plethora of solutions enabled by blockchains have changed how we imagine applications in significant domains such as finance, gaming, the sharing economy, etc. In addition to transforming the underlying architecture of today's Internet, blockchain technology has also led to several innovations in the field of cryptography to enable fully privacy-preserving applications that directly solve the issues of data breaches and targeted advertisements currently plaguing the Internet.

This transformative technology, however, brings along its challenges; it suffers from a *Trilemma*, which states that it is hard for a blockchain to be decentralized, scalable, and secure simultaneously. This is called the Scalability Trilemma, and all blockchains need to address it.

This thesis constitutes a systematic investigation of the Trilemma. It documents the trade-offs in designing decentralized applications, investigates existing ones, and suggests mitigation approaches. The work is split into three domains: the sharing economy, energy, and finance.

The first two results in this thesis are evidence of the Trilemma: the first one focuses on a decentralized car-sharing application on Ethereum and uncovers its scalability shortcomings; the second investigates a popular decentralized storage application and reveals a security vulnerability.

The next three results focus on the energy domain: the first investigates the applicability of a frequently suggested scaling solution to some prominent use cases in the energy domain. We conclude that the use case of peer-to-peer energy trading cannot be scaled using existing scaling solutions deployed on permissionless blockchains. Hence, by utilizing the lessons learned in this thesis, the second and third studies propose fully privacy-preserving and decentralized billing models for peer-to-peer energy trading.

Lastly, focusing on the financial domain and, more specifically, markets with ephemeral participating relationships, this thesis provides a modular approach to addressing the Scalability Trilemma by proposing a trustless overlay network allowing global payments using cryptocurrencies in a retail setting with an overall delay of less than one second.

We hope the lessons learned in this thesis help make decentralized applications more scalable and secure without compromising on any aspect, and that it inspires anyone who reads it to become a strong advocate of decentralization.

Beknopte samenvatting

Tegenwoordig is een leven zonder internet ondenkbaar: we gebruiken het om in contact te blijven met onze sociale kring, onze droombaan te vinden, onze mening te uiten en onze favoriete documentaires te bekijken; we leven ons tweede, efficiëntere en meer verbonden leven online. Gezien de jaren van innovatie die het internet heeft ondergaan, zijn er meer use cases ontstaan, allemaal met één gemeenschappelijke architecturale benadering, namelijk een gecentraliseerde architectuur. Centralisatie van diensten heeft geleid tot een naadloze gebruikerservaring, wat geresulteerd heeft in de enorme groei van internetgiganten zoals Meta, Google, enz.

De financiële motor van gepersonaliseerde advertenties drijft de groei van het internet aan. Ons tweede leven op het internet heeft ons ook tot doelwit gemaakt van op maat gemaakte advertenties, die mogelijk worden gemaakt door de grootschalige dataverzameling over al onze handelingen. Alsof gerichte advertenties nog niet erg genoeg zijn, zijn datalekken die jaarlijks gevoelige informatie over miljarden mensen onthullen een andere tekortkoming van deze gecentraliseerde architectuur.

Deze problemen hebben geleid tot een klinkende roep om decentralisatie, met blockchain technologie in de voorhoede van deze beweging. Oplossingen zoals Bitcoin en Ethereum hebben de manier veranderd waarop we ons toepassingen voorstellen in belangrijke domeinen zoals financiën, gaming, de deeleconomie, enz. Naast het transformeren van de onderliggende architectuur van het huidige internet, heeft blockchain technologie ook geleid tot verschillende innovaties op het gebied van cryptografie om volledig privacy-vriendelijke toepassingen mogelijk te maken die een directe oplossing bieden voor de problemen van datalekken en gerichte advertenties die het internet momenteel teisteren.

Deze transformatieve technologie brengt echter ook uitdagingen met zich mee; ze lijdt aan een *Trilemma*, dat stelt dat het moeilijk is voor een blockchain om tegelijkertijd gedecentraliseerd, schaalbaar en veilig te zijn. Dit wordt het Schaalbaarheidstrilemma genoemd en alle blockchains moeten dit probleem aanpakken.

Dit proefschrift is een systematisch onderzoek van het Trilemma. Het documenteert

de afwegingen bij het ontwerpen van gedecentraliseerde toepassingen, onderzoekt bestaande toepassingen en stelt oplossingen voor. Het werk is opgesplitst in drie domeinen: de deeleconomie, energie en financiën.

De eerste twee resultaten in dit proefschrift illustreren het Trilemma: het eerste richt zich op een gedecentraliseerde autodeeltoepassing op Ethereum en onthult de tekortkomingen in de schaalbaarheid; het tweede onderzoekt een populaire gedecentraliseerde opslagtoepassing en onthult een beveiligingslek.

De drie volgende resultaten richten zich op het energiedomein: het eerste onderzoekt de toepasbaarheid van een populaire schalingsoplossing op enkele prominente use cases in het energiedomein. We concluderen dat een vaak vermeld use case, namelijk peer-to-peer energiehandel, niet kan worden geschaald met bestaande schalingsoplossingen die worden ingezet op publieke rechtenloze blockchains. Met behulp van de lessen uit dit proefschrift stellen het tweede en derde onderzoek volledig privacy-vriendelijke en gedecentraliseerde facturatiemodellen voor peer-to-peer energiehandel voor.

Tot slot, gericht op het financiële domein en, meer specifiek, markten met kortstondige deelnemers, biedt die proefschrift een modulaire benadering voor het aanpakken van het Schaalbaarheid Trilemma door een vertrouwensloos overlay netwerk voor te stellen dat wereldwijde betalingen mogelijk maakt in een retail omgeving met een totale vertraging van minder dan een seconde.

We hopen dat de lessen die we geleerd hebben in dit proefschrift helpen om gedecentraliseerde applicaties schaalbaarder en veiliger te maken zonder enig compromis te sluiten, en dat het iedereen die het leest inspireert om een groot voorstander van decentralisatie te worden.

List of Abbreviations

ATM Automated Teller Machine

B2B Business-to-Business

B2C Business-to-Consumer

BFT Byzantine Fault Tolerance

C2C Consumer-to-Consumer

CKG Collaborative Key Generation

DAG Directed Acyclic Graph

DHT Distributed Hash Table

DLT Distributed Ledger Technology

DoS Denial-of-Service

DPoS Delegated Proof-of-Stake

EOA Externally Owned Accounts

EV Electric Vehicle

EVM Ethereum Virtual Machine

FHE Fully Homomorphic Encryption

GENs Electricity Generators

HEMS Home Energy Management System

HTLC Hashed Time Lock Contract

IPFS Interplanetary File System

KYC Know-Your-Customer

MB MegaByte

MPC Multi-Party Computation

NC Nakamoto Consensus

NFT Non-Fungible Token

NIZK Non-Interactive Zero-Knowledge Proof

P2P Peer-to-Peer

PBFT Practical Byzantine Fault Tolerance

PoA Proof-of-Authority

PoB Proof-of-Burn

PoC Proof-of-Capacity

PoET Proof-of-Elapsed-Time

PoS Proof-of-Stake

PoST Proof-of-Space-Time

PoW Proof-of-Work

PRF Pseudo-Random Function

RES Renewable Energy Sources

RM Retail Market

RP Retail Price

rrTE Randomness-Reusable Threshold Encryption

RSA Rivest Shamir Adleman

SC Smart Contract

SM Smart Meter

SNIPPET Secure and Privacy-Friendly Peer-to-Peer Electricity Trading

TP Trading Price

tps Transactions Per Second

TSPS Threshold Structure Preserving Signatures

UI User Interface

UTXO Unspent Transaction Output

ZKP Zero-Knowledge Proof

Contents

Abstract vii

Beknopte samenvatting ix

List of Abbreviations xiii

Contents xv

List of Figures xxi

List of Tables xxv

I Prelude 1

1 Introduction 3
1.1 Motivation . . . 3
1.2 Contributions . . . 6
1.3 Outline . . . 7

2 Decentralization 9
2.1 Decentralization in Practice . . . 10
2.1.1 Infrastructure . . . 11
2.1.2 Network Topology . . . 12
2.1.3 Authority . . . 12
2.2 Blockchain Technology . . . 13
2.2.1 Properties of a Blockchain . . . 14
2.2.2 Ethereum . . . 21

3 The Scalability Trilemma 27
3.1 Categorizing Blockchains . . . 27

3.2 Approaches to Addressing the Trilemma . . . 29
3.3 The Trilemma in Practice . . . 34
3.3.1 Sharing Economy . . . 34
3.3.2 Energy Domain . . . 37
3.3.3 Financial Domain . . . 44

4 Conclusion and Future Work **51**
4.1 Conclusion . . . 51
4.2 Discussion . . . 52
4.3 Future Work . . . 53

Bibliography **57**

II Publications **71**

SC2Share: Smart Contract for Secure Car Sharing **75**
1 Introduction . . . 77
2 Background . . . 79
2.1 SePCAR . . . 79
2.2 Smart contracts in ethereum . . . 79
3 System model, threat model and design requirements . . . 80
3.1 System model . . . 81
3.2 Threat model and assumptions . . . 81
3.3 Design requirements . . . 81
4 SC2Share . . . 83
4.1 Overview of SC2Share . . . 83
4.2 SC2Share in detail . . . 84
5 Evaluation . . . 86
5.1 Functionality analysis . . . 87
5.2 Security and privacy analysis . . . 88
5.3 Deployment and usage cost . . . 89
6 Conclusions and future work . . . 91
References . . . 91

Exploring the Storj Network: a Security Analysis **95**
1 Introduction . . . 98
2 Incentivization-driven distributed file storage systems . . . 99
2.1 Overview . . . 99
2.2 Related work . . . 100
3 Storj architecture . . . 101
3.1 Services . . . 101
3.2 System behavior . . . 102

4 Security evaluation . . . 108
4.1 DoS vulnerability in Storj's dev./test environment . . . 108
4.2 Experimental security evaluation . . . 111
5 Discussion . . . 112
6 Conclusions . . . 112
References . . . 113

Off-chain State Channels in the Energy Domain **117**
1 Introduction . . . 119
2 Background and Related Work . . . 120
2.1 Blockchain and Scalability . . . 120
2.2 State Channels . . . 121
2.3 Related Work . . . 122
3 Methodology . . . 123
3.1 Application Costs . . . 123
3.2 Methodology . . . 123
4 Evaluation of Use Cases in the Energy Domain . . . 124
4.1 Energy Trading . . . 124
4.2 Local Energy Trading . . . 126
4.3 Energy Certificate Trading . . . 126
4.4 Electricity Consumption Optimization . . . 127
4.5 Electric Vehicle Charging . . . 128
4.6 Tariff Decisions . . . 129
5 Conclusion . . . 129
References . . . 129

Billing Models for Peer-to-Peer Electricity Trading Markets with Imperfect Bid-Offer Fulfillment **133**
1 Introduction . . . 135
2 Billing Models . . . 137
2.1 Billing Model for Retail Markets – the status quo . . . 137
2.2 Billing Models for P2PM with RM as Back-up . . . 139
2.3 Billing Models for P2PM with Mid-Market as Back-up . . . 142
3 Evaluation . . . 143
3.1 Use-case . . . 143
3.2 Data Generation . . . 143
3.3 Simulation Results . . . 145
4 Conclusions and Future Work . . . 146
References . . . 146

Fully Privacy-Preserving Billing Models for Peer-to-Peer Energy Trading **149**
1 Introduction . . . 151
2 Related work . . . 153

3 Preliminaries . . . 155
3.1 System Model . . . 155
3.2 Threat Model and Assumptions . . . 156
3.3 Design Requirements . . . 156
3.4 Billing Models . . . 157
3.5 Cryptographic Primitives . . . 160
4 Privacy-enhanced billing models . . . 161
4.1 Overview . . . 162
4.2 Setup Phase . . . 162
4.3 Client Setup for Billing Model with Universal Cost Split . . . 163
4.4 Server's Billing Model with Universal Cost Split . . . 164
4.5 Bill settlement . . . 165
5 Evaluation . . . 166
5.1 Security Analysis . . . 166
5.2 Theoretical analysis . . . 167
5.3 Simulations . . . 168
6 Conclusions and Future Work . . . 170
References . . . 171

Reusable, Instant and Private Payment Guarantees for Cryptocurrencies 177
1 Introduction . . . 179
2 Our construction . . . 182
2.1 Design Motivation . . . 182
2.2 Strawman Construction . . . 182
2.3 Our solution . . . 184
3 Preliminaries and Formal Construction . . . 188
3.1 NIZK Languages . . . 192
3.2 Security Analysis . . . 192
4 An Efficient Instantiation . . . 198
5 Performance Analysis . . . 200
6 Discussion . . . 203
7 Conclusion . . . 206
References . . . 207

A Appendix 213
A.1 Cryptographic Primitives . . . 213
A.1.1 Pseudo-Random Function (PRF) . . . 214
A.1.2 Shamir Secret Sharing . . . 215
A.1.3 Digital Signatures and TSPS . . . 215
A.1.4 Commitment schemes . . . 220
A.1.5 Threshold Encryption . . . 221
A.1.6 Collaborative Key Generation and RRTE . . . 223
A.1.7 Non-Interactive Zero-Knowledge proofs . . . 224

A.1.8 NIZK Language Realization 229

List of Figures

1.1 Outsourcing of responsibilities in C2C markets 5

2.1 A network topology allowing for decentralization 9
2.2 Bob pays Alice 1 Bitcoin 17
2.3 Digital signature in Bitcoin 18
2.4 Bitcoin transaction input and output 22
2.5 The Ethereum State Transition function is applied on each block. The addresses 14c5f8ba & 4096ad65 correspond with EOA, whereas the addresses bb75a980 & 892bf92f correspond with contract accounts . . 24

3.1 A payment channel network routing a payment between Alice and Charlie via Bob and ensuring atomic payments by using Hashed Time Lock Contracts (HTLC) 32
3.2 A payment channel hub routing a payment between Alice and Charlie and ensuring atomic payments by using an arbitrary puzzle (e.g., RSA) puzzles) 32
3.3 Sc2Share in the Scalability Trilemma 35
3.4 Storj in the Scalability Trilemma 36
3.5 Our billing models in the Scalability Trilemma 43
3.6 Timeline of a transaction in our overlay network 48
3.7 Our overlay network in the Scalability Trilemma 48
3.8 Construction of our overlay network 49
3.9 Proactive double-spending detection 50

SC2Share: Smart Contract for Secure Car Sharing **75**
1.1 System model of SC2Share 80
1.2 Overview of the SC2Share system 82
1.3 List of booking details 84

Exploring the Storj Network: a Security Analysis **95**

2.1 Network level view of Storj . . . 101
2.2 Creating a bucket on a satellite . . . 103
2.3 The Storj erasure numbers . . . 104
2.4 Initial upload of encrypted path and optional expiration date . . . 104
2.5 Uploading an inline segment to the satellite . . . 105
2.6 Contact satellite before uploading a remote segment . . . 105
2.7 Uploading one piece to a storage node . . . 106
2.8 Sending metadata of a remote segment to the satellite . . . 107
2.9 The auditing process scoped to one stripe . . . 108
2.10 A DoS attack orchestrated by one malicious storage node . . . 109

Off-chain State Channels in the Energy Domain **117**
3.1 In this figure n participants create a state deposit by sending on-chain txs (lock). After this they can update the state deposit off-chain, which is illustrated with u_n. At the end they close the channel and unlock the deposit . . . 122
3.2 In this figure we depict how our procedure is applied to classify the applicability of state channels . . . 125

Billing Models for Peer-to-Peer Electricity Trading Markets with Imperfect Bid-Offer Fulfillment **133**
4.1 P2P market with RM as a fall-back option . . . 139
4.2 Suppliers' trading volumes at RT and P2P billing models . . . 144
4.3 Suppliers' profits at RT and P2P billing models . . . 144
4.4 Difference in Bills/Rewards of all participants . . . 145
4.5 Difference in Bills/Rewards of P2P participants . . . 145
4.6 Difference in Bills/Rewards of non-P2P participants . . . 146

Fully Privacy-Preserving Billing Models for Peer-to-Peer Energy Trading **149**
5.1 P2P market with RM as a fall-back option . . . 158
5.2 Overview of our privacy-preserving billing model . . . 162
5.3 Average computation time for all households and billing periods . . . 171
5.4 Computation times per billing period for 50 households . . . 171
5.5 Computation times per billing period for 100 households . . . 172
5.6 Computation times per billing period for 150 households . . . 172

Reusable, Instant and Private Payment Guarantees for Cryptocurrencies **177**
6.1 Snappy's construction and a naive attempt to provide protocol-level privacy . . . 183
6.2 Timeline of a transaction . . . 185
6.3 Our construction . . . 186
6.4 Proactive double-spending detection . . . 187
6.5 Our Construction . . . 190

6.6 Security Games 194
6.7 Experimental setup. Dotted lines and shapes represent parties and workload that do not affect the leftmost customer or the leftmost Merchant. Our experiment only considers the solid lines and shapes. From the perspective of the leftmost client and the leftmost merchant, this is equivalent to running the full system. While statekeepers are the same as merchants, we make these two sets distinct for clarity 201
6.8 Latency comparison. Total transaction time for 1000 tx per second vs the number of statekeepers 202

A.1 Indexed Diffie-Hellman Message Space in the ROM 214
A.2 Static Semantic Security Experiment 223
A.3 Proving the knowledge of DLog 227
A.4 To prove a multiplication relation between committed values 227
A.5 Proving well-formedness of the DY's PRF 228

List of Tables

1.1 Mapping of our contributions to each paper. 8

SC2Share: Smart Contract for Secure Car Sharing **75**
1.1 Deployment costs of SC2Share. 88
1.2 Costs of executing each transaction in SC2Share. 89

Exploring the Storj Network: a Security Analysis **95**
2.1 Incentivization-driven distributed storage systems. 99

Off-chain State Channels in the Energy Domain **117**
3.1 Comparison of blockchain-based energy use cases according to defined criteria. 130

Billing Models for Peer-to-Peer Electricity Trading Markets with Imperfect Bid-Offer Fulfillment **133**
4.1 **Abbreviations and Notations** 138

Fully Privacy-Preserving Billing Models for Peer-to-Peer Energy Trading **149**

Reusable, Instant and Private Payment Guarantees for Cryptocurrencies **177**
6.1 Costs of transactions on our smart contract deployed on Ethereum, and the cost as a proportion of a standard transaction $\Delta = 21\,000$. 203

Part I

Prelude

Chapter 1

Introduction

> If you have trust issues, why not harness them for a career in distributed systems?
>
> Random Tweeter

1.1 Motivation

Browsing the Internet to find a recipe or check what is happening in your friends' lives (e.g., Google, Meta), paying a friend through a mobile application (e.g., Cash App), renting an apartment for a vacation (e.g., Airbnb) is something that we all often do. These services' ease of use, user-friendliness, and efficiency are significant reasons for their widespread adoption: Meta serves around 3 billion active users monthly as of the time this thesis was written [9].

The advent of the Internet and its rapid adaptation has significantly contributed to today's economy. This can be seen by the immense growth of the global e-commerce market; these markets are split into three major categories, namely, Business-to-Business (B2B), Business-to-Consumer (B2C), and Consumer-to-Consumer (C2C) markets. A significant difference is their client base: B2B markets are commerce models between businesses, B2C markets are commerce models between a business and a customer, and C2C markets are commerce models between individual sellers and customers. Due to their e-commerce nature, all these market models involve an online

platform for sales that can either be self-hosted (B2C) or hosted by an intermediary (B2B, C2C).

Aided by the widespread reach the Internet has, e-commerce markets have seen rapid growth, and several firms predict a 9.7% growth of the B2C e-commerce market by 2028, resulting in a total market capitalization of $7,656.45 billion [3]. From the plethora of available C2C platforms, the most prominent ones include Etsy, eBay, Aliexpress, and Amazon. Etsy alone has seen a significant increase in active buyers from 9.3 million in 2012 to 96.34 million in 2021 [7]. Growth at this scale has also been observed in other popular C2C platforms. Aliexpress and Amazon were in the top 5 most visited online retail websites in 2022 [12], with Amazon's German marketplace being Europe's most visited e-commerce website [8].

A common occurrence and the major factor underlying the efficiency in current C2C platforms is a centralized *intermediary*. In current C2C models, users typically outsource the network upkeep, storage, and monetary responsibilities to such intermediaries, similar to Fig. 1.1. Sellers (resp. buyers) offer (resp. purchase) services or products on platforms hosted by these intermediaries. Upon finding a suitable service or product, the buyer contacts the seller through the same platform, and once an agreement is reached, settlement also happens through the same intermediary.

The presence of such intermediaries abstracts most of the cumbersome responsibilities away from the other parties, making these platforms popular. This comfort, of course, comes at a cost. Intermediaries usually have different revenue models.

On the one hand, C2C platforms such as Airbnb charge a fee (14-16% of the booking amount), which can be either split between the guest and host or solely borne by the host [16]. On the other hand, social media platforms such as Meta and search engines such as Google are free to use. The quote "There ain't no such thing as a free lunch" has been around for decades, presumably starting from the time when American pubs offered free food to customers who would buy drinks, and popularized by Heinlein in his book "The Moon is a Harsh Mistress" [64]. Meta owns 18 data centers around the world, totaling over $20 billion [48] in operational costs. Thus, one must wonder: how does using Meta incur no cost to the end user? The answer lies in what Meta stores in these data centers.

Every time a user logs into their Facebook account, interacts with posts, or visits the Facebook marketplace to browse products for sale, Meta collects and stores this browsing data. The data is then used by a team comprising expert data scientists, who analyze it and mine 'useful information' from it. This information is, at a high level, a digital twin of the user and helps increase their engagement with various content. Meta uses this information to target advertisements to its users to gain ad revenue from advertisers. In 2021, Meta's ad revenue was nearly $115 billion [10]. It is not just Meta; companies such as Google that date back to the Web 1.0 era presented an ad

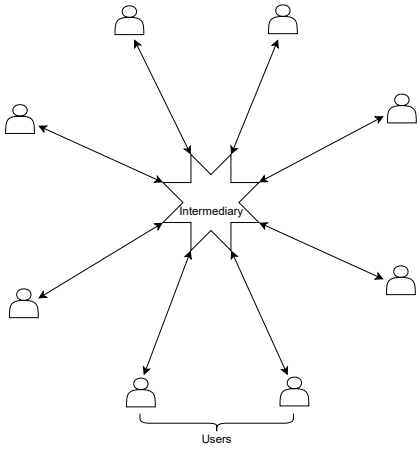


Figure 1.1: Outsourcing of responsibilities in C2C markets

revenue of $209.49 billion in 2021 through their free search engine [11].

The World Wide Web, or in retronym, Web 1.0, was conceived by Sir Tim Berners-Lee. Web 1.0 was mainly read-only and provided ways to publish and access documents online. However, in the early 2000s, a paradigm shift took place — users became integral to the evolution of the World Wide Web as the technology now enabled the Internet to host media content. The number of contributing users grew, and technological advances in broadband and server technologies enabled 'heavier' content, such as photos and videos, to be contributed, leading to the inception of websites such as YouTube and Flickr. This new paradigm became known as Web 2.0 or the read-write web [61]. In addition to the subscription-based model used by popular websites in the era of Web 1.0, the shift to Web 2.0 introduced various new business models. These business models were based on the presence of a centralized intermediary for efficiency and, as a result, provided them with lucrative incentives. These incentives are what enabled the continuous growth of the Web and the services that it offers. Imagining work or personal life without these services has now become hard.

As mentioned earlier in this section, intermediaries in Web 2.0 make these services user-friendly and efficient. The intuitive User Interface (UI) offered by most websites plays in the favor of both the users and intermediaries. The easier the UI, the more click data it generates. More clicks translate to more data collected, leading to subjective advertisements targeted to users. Consequently, more accurate advertisements translate to more money for intermediaries. Hence, none of the services offered on the Internet is free. Some services are paid for upfront, whereas others monetize *privacy*. Such business models may be profitable for the intermediaries, but they lead to many

problems. For example, Target, a B2C giant, revealed to a teenager's parents that she was pregnant through targeted advertisements before she could tell them herself [65].

By being 'warehouses of data', such companies also sell the data they collect to various firms, of which the now insolvent Cambridge Analytica is an infamous example. The field of big data encompasses the science used to mine such data warehouses and obtain helpful information that companies eventually use for making large profit margins: data is the new oil of the digital economy. In addition to profit margins, firms like Cambridge Analytica are also examples of how the problem of centralized data collection is beyond money, and can extend to psychological targeting. However, targeted ads and capitalizing on data are not the only ways these data warehouses aid in breaching our privacy.

The intermediaries often also become a single point of failure: one of the most significant data breaches in recent years was from an adult site known as CAM4. This data breach consisted of approximately 10.9 billion data records [13], including sensitive information such as sexual orientation. Data breaches are a common theme among these intermediaries, with many popular services such as LinkedIn, Yahoo, and Uber, unfortunately, having been breached several times already [13, 1]. Hence the famous saying, "Big Data = Big Breaches."

Other issues of this reliance on centralized architectures are availability and censorship. October 2022 was a bad month for Meta [2], with its most significant services, such as WhatsApp and Instagram, going down. There is no certified public measurement of the impact these outages have caused on businesses and people; however, by relying on social media platforms for marketing and WhatsApp for communication, the impact is obvious.

Although a controversial topic, it should be pointed out that censorship is also prevalent in these intermediary-based architectures. One prominent example of censorship is 'Deplatforming', showcased with the banning of Donald Trump's profile by Twitter (now known plainly as X). The issue of censorship stemming from centralization has been around for decades, and the concept of decentralization is often seen as its solution. The captivating enticement for decentralization of the Internet can be dated back to the late 90's, an example of which is RJ Anderson's Eternity Service [17].

All issues pointed out so far in this section arising from these intermediaries becoming a single point of failure regarding privacy, availability, and censorship beg the question: Is decentralization a silver bullet to mitigate all of them?

1.2 Contributions

This thesis contains the following contributions:

C1 *An implementation of existing centralized solutions in a decentralized manner, uncovering the trade-offs required for decentralization*: In [83], we proposed a decentralized alternative for P2P car sharing. Our implementation of Sc2Share highlighted an existing problem in blockchains, i.e., the Scalability Trilemma. More specifically, it shows that security and decentralization are achieved at the cost of scalability. In "Billing Models for Peer-to-Peer Electricity Trading Markets with Imperfect Bid-Offer Fulfillment" [84], we implement novel P2P billing models as a decentralized alternative to existing billing models.

C2 *Analyses of proposed decentralized services in various domains*: We used the lessons learned from implementing Sc2Share and analyzed other proposed decentralized services in various domains. In "Exploring the Storj Network: a Security Analysis" [46], a security analysis of a popular decentralized storage network is performed, and a Denial-of-Service (DoS) vulnerability is found. In "Off-chain state channels in the energy domain" [31], the scalability of proposed decentralized applications in the energy domain is explored. In "Instant, Reusable and Private Payment Guarantees for Cryptocurrencies" [82], we analyzed existing state-of-the-art solutions that offer fast cryptocurrency payments in retail markets and highlighted their shortcomings.

C3 *Improved existing decentralized counterparts based on the analysis of their weaknesses*: Based on the analysis of scalability in various sub-domains of the energy market, it was observed that performing billings and settlements in local energy trading markets is not scalable when deployed on a public blockchain. Hence, we introduce a stratified decentralized architecture to design novel billing protocols for P2P energy trading in our paper "Billing Models for Peer-to-Peer Electricity Trading Markets with Imperfect Bid-Offer Fulfillment" [84]. Based on our analysis of security and privacy requirements in P2P electricity markets in [89], we later proposed a fully privacy-preserving variant of the billing model with universal cost split in our paper "Fully Privacy-Preserving Billing Models for Peer-to-Peer Electricity Trading Markets". The final contribution of this thesis is an overlay network published in "Instant, Reusable and Private Payment Guarantees for Cryptocurrencies" [82]. This overlay network is capable of providing instant payment settlements. This is useful, e.g., in retail markets accepting payments in cryptocurrencies.

We map the contributions of our papers in Table 1.1

1.3 Outline

This thesis is split into two parts. Part I is the "Prelude", and describes the motivation and contributions of this thesis along with giving the necessary background.

Table 1.1: Mapping of our contributions to each paper.

Paper	C1	C2	C3
SC2Share [83]	✓		
Exploring Storj [46]		✓	
State Channels in Energy Domain [31]		✓	
Billing Models in P2P Electricity Trading [84]	✓		✓
Fully Privacy-Preserving Billing Models			✓
Instant, Reusable and Private Payment Guarantees for Cryptocurrencies [82]		✓	✓

Rest of this part is structured as follows: Chapter 2 discusses what decentralization is and then explains blockchain technology in detail. After that, in Chapter 3, we discuss how blockchains can be categorized, followed by explaining the Scalability Trilemma and various solutions in the literature to circumvent it. Later, it is explained how the work done in this thesis revolves around the Scalability Trilemma. Finally, Chapter 4 concludes this part and lists several concrete future studies.

Part II contains the following publications related to this thesis:

1. Sc2share: Smart contract for secure car sharing, a decentralized car-sharing service deployed on Ethereum [83] —-See p. 75.
2. An architectural overview and security analysis of Storj [46] —-See p. 95.
3. A methodology to assess use-cases in the Energy domain and the applicability of State Channels to scale them [31] –See p. 117.
4. Novel billing models for P2P energy trading that consider discrepancies in committed and utilized volumes of traded electricity [84] —-See p. 133.
5. A fully privacy-preserving billing model for P2P trading markets —-See p. 149.
6. An overlay network to enable reusable, instant, and private payment guarantees for cryptocurrencies [82] —-See p. 177.

The complete list of publications can be found on page 73.

Chapter 2

Decentralization

Evident from the word itself, decentralization is the opposite of the intermediary-based architecture. It is the subset of a distributed architecture, i.e., a system with several spatially separated components coordinated via message passing. Unlike a distributed architecture that may be controlled by an intermediary, a decentralized architecture is controlled by multiple entities [115]. The idea of decentralization can be observed in the city-states of Ancient Greece. Each city-state had its own government and laws, leading to a more democratic form of governance. Due to direct democracy, citizens had a more significant influence on decisions that affected their lives.

In the digital era that we live in, issues stemming from centralization have given rise to a more robust demand for more control over our data. Recent specifications such as Solid that enable its users to store data in decentralized data stores called *Pods* [103] is an example of such a demand. Peer-to-Peer (P2P) file-sharing services such as

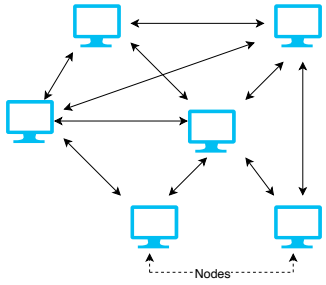


Figure 2.1: A network topology allowing for decentralization

Napster [120] evolved to a decentralized architecture when Napster was shut down for distributing pirated music. This ban proved relatively easy to enforce for the authorities due to the original architecture of Napster, which relied on a centralized repository to keep track of all files being shared. This led to the development of more robust P2P file-sharing services that utilize a decentralized architecture, such as BitTorrent [40].

The call for decentralization is not just limited to personal data security or circumventing censorhip. The inception of Bitcoin [91] started a revolution that has seen this demand (and subsequent adoption) in many verticals of the world's infrastructure, starting from finance and extending to art, science, real estate, and governance.

In 2009, a post by *Satoshi Nakamoto* made waves on the P2P foundation forum[1]. As stated by Satoshi Nakamoto himself, the motivation behind Bitcoin was to challenge the abundance of trust we place in centralized intermediaries (in the case of Bitcoin: banks), to improve our privacy, secure us against identity theft, and preserving our money. However, this trust is often breached, as evidenced by events like the 2008 market crash. By enabling secure monetary transactions to anyone in the world without needing a centralized intermediary, Bitcoin opened the floodgates for pouring innovation in the field of decentralization and became ubiquitous as its catalyst.

Decentralization, however, is often confused with a distributed architecture. Ensuring that nodes participating in the network are spatially separated does not mean the architecture is decentralized. Entities like Meta and Google commonly use this approach to ensure the robustness of their systems. We still trust that entity as long as the same entity controls such spatially separated nodes. What does it then mean for a service to be decentralized?

Decentralization is achieved when no single authority/entity provides the entire infrastructure required to run a service. BitTorrent and Bitcoin are examples of such an architecture. Although used to run different applications, neither of these services relies on a single authority/entity to provide the necessary infrastructure. As shown in Fig. 2.1, nodes (or participants) in a decentralized architecture communicate directly with each other to execute a service over which all have a portion of the control.

2.1 Decentralization in Practice

As we show in this thesis, there is no one-size-fits-all approach to designing decentralized systems. As detailed by Troncoso et al. in [115], decentralized systems that are deployed in practice differ in three major aspects:

[1] https://p2pfoundation.ning.com/forum/topics/bitcoin-open-source?id=2003008%3ATopic%3A9402&page=4#comments

- Infrastructure: which nodes provide the necessary means to maintain the decentralized service;
- Network Topology: how are the different nodes organized in the network to route traffic;
- Authority: the distribution of governance authority roles throughout the network.

2.1.1 Infrastructure

Decentralized services differ in terms of infrastructure; specifically, who is providing the infrastructure? One type of system is where the service's users directly maintain a group of nodes. An example of such a system is Bitcoin. Conversely, there are systems such as R3 Corda [30] where a non-user, non-collaborating, untrusted group of third parties maintain the system. There are various combinations of both, allowing for different trade-offs.

A decentralized system is said to have a *user-based infrastructure* if a group of nodes maintains the service by contributing resources (primarily digital, such as storage and bandwidth) while simultaneously consuming them. An example of this infrastructure is a permissionless blockchain (e.g., Bitcoin, Ethereum [121]), where a collective of nodes maintains the distributed ledger by contributing their computational resources. More importantly, these nodes are also rewarded in the native cryptocurrency as an incentive to fulfill their responsibilities, hence simultaneously making them operators of this service.

Alternatively, a decentralized system is said to have a *user-independent infrastructure* if a group of nodes maintains the system by contributing resources without being its users. An example of such infrastructure is a permissioned blockchain (e.g., Hyperledger Fabric [19], R3 Corda [30]), where a set of known, predetermined third-parties maintain the distributed ledger. These third parties are not necessarily trusted and are assumed to be non-colluding. A famous example of a permissioned blockchain in practice is the KUBE project[2], using Hyperledger Fabric to streamline the Know-Your-Customer (KYC) process for four central Belgian banks.

Hybrid decentralized systems combining user-based and user-independent infrastructures also exist. A famous example of this infrastructure is Tor [49]. Nodes that utilize Tor as a service also run its relays; however, a known set of servers are responsible for operating the Directory Authorities that keep track of all healthy relays.

[2] https://www.kube-kyc.be/en/

2.1.2 Network Topology

Decentralized networks may differ in network topology, i.e., how the nodes are organized to route traffic.

Directed Acyclic Graph (DAG) based solutions such as MeshCash [25] and SpaceMesh [26] use a *Mesh topology*. Nodes arranged in a mesh route messages to other nodes in the system either through *gossiping* or by *flooding* messages. In a gossip protocol, each node selects a subset of nodes that they are connected to and sends them messages. Alternatively, nodes can use a flooding protocol to send messages to all neighboring nodes. Mesh topologies are popular for being robust in unstable network connections.

Another topology that decentralized systems utilize is based on Distributed Hash Tables (DHTs). In such topologies, nodes maintain information about their neighbors in the form of tables and *greedily* send information to the nodes closer to the intended recipient. Such topologies are usually efficient but do not preserve privacy. BitTorrent is a famous example of a decentralized system that uses DHTs.

Some decentralized systems also utilize *Super-nodes* in their network topologies. Such nodes contribute more resources to the system and have more traffic routed through them. These topologies are usually efficient and support high network throughput since they are well connected; however, these nodes may become a single point of failure regarding availability and integrity. In popular payment channel networks such as the Lightning Network [99], super nodes emerge naturally [126] due to some nodes committing more resources (e.g., monetary collateral).

Finally, systems such as Tor utilize a *Stratified* topology. In such topologies, some nodes have a unique role, for instance, Directory Authorities in the Tor network. Payment channel hubs such as TumbleBit [63] also utilize a special hub operator to route payments from a sender to a receiver, hence utilizing a stratified topology.

2.1.3 Authority

Decentralized systems utilize the notion of authority in a very flexible way; hence, even though no single node in the network can control the entire service, in some cases, decentralized systems may have some notion of authoritarian roles.

When nodes interact directly, without any notion of authority, the decentralized system is called *ad hoc*. Ad hoc systems require participating nodes to perform all actions necessary to keep the service running. BitTorrent is classified as an ad-hoc system since there are no clearly defined roles of authority.

In P2P systems such as public blockchains, there are some nodes with more authority compared to others. For example, nodes that play the role of *miners* provide intensive computation power to keep the network running in return for rewards. Additionally, some nodes that act as *archives* have authority since they store the entire blockchain history. Crucially, these roles can be played by any node willing to contribute the resources necessary to benefit the network. Hence, in such decentralized systems, *nodes assist other nodes*.

Some decentralized networks may be *federated*, where nodes trust specific provider nodes to play the role of authorities. In blockchains that utilize the delegated proof-of-stake (DPoS) consensus, all participating nodes elect and delegate their stake (monetary deposits) to such elected nodes. These elected nodes act as *delegates* and maintain the system. Each delegate is responsible for their delegatees since their misbehavior would result in losing their delegatees' stake. An example of a DPoS blockchain in practice is the EOSIO [59] network, where 21 delegates are elected by other nodes with sufficient stake in the network.

Usually, in decentralized systems, when some nodes gain more authority than others, such as in Bitcoin (naturally) or DPoS (by design), safeguards are put in place to keep them accountable. These mechanisms involve transparency of these nodes' operations to catch malicious actions. Consequently, such nodes are also incentivized to behave honestly, primarily by monetary means.

A decentralized system may be deployed using any infrastructure or network topology and can have any notion of authority as long as multiple independent participants maintain the service it provides. Compared to the era of R.J. Anderson's Eternity Service [17], the development of an interest in decentralized applications has seen a noteworthy increase among researchers and the general public.

2.2 Blockchain Technology

The Web 3.0 revolution has taken many verticals of the world's infrastructure by storm. Fields such as finance, governance, etc. have all seen rapid innovation. The credit for this revolution can only be taken by blockchain technology, the distributed ledger that enables applications such as Bitcoin.

Folklore suggests that Bitcoin's *white paper* [91] is what gave birth to the now popularized distributed ledger, also known as a *blockchain*. However, this is not the case. Already in 1990, Haber and Stornetta envisioned the idea of a blockchain [60]. Their work proposed a solution to authenticating documents by time-stamping them using a time-linked chain. The first commercial deployment of a blockchain dates back

to 1994 when Haber and Stornetta co-founded Surety Technologies to offer digital time-stamping services [27, 66].

As we advance, this thesis will use Bitcoin and Ethereum as instructive examples of how blockchains work. Note that this thesis only discusses the concept of Blockchain at a high level to familiarize the readers with the technology; however, deep technical details, such as network layer properties, etc., are omitted. Interested readers are referred to Narayanan et al.'s book [92] and the Ethereum yellow paper [122] for a complete view of these systems.

Before we discuss Bitcoin, we first describe the modern banking system. Banks can be classified into various types, such as central, commercial, retail, etc.

A retail bank is a sub-classification of banks with which most of us interact daily. Each retail bank client has an account history maintained by the bank that comprises all debit/credit-related information of their bank accounts. These account histories are used for various purposes, such as calculating a client's credit scores, searching for possible money laundering via automated tools, etc.

The modern banking system works very well, yet it places a lot of power in the hands of a small number of organizations. Bitcoin's idea was to decentralize this power and let the masses regain control.

More specifically, instead of a centralized bank controlling the account history (usually done in the form of a ledger), all Bitcoin network participants collectively maintain the account history of all users in the Bitcoin network. Thus, instead of a centralized ledger, Bitcoin utilizes the concept of a *decentralized* ledger controlled by anyone interested in participating. Anyone can use Bitcoin to pay or receive money from anyone else without restrictions.

2.2.1 Properties of a Blockchain

An established way to maintain this decentralized ledger is using a **Blockchain**.

Let B denote a blockchain, represented as a sequence of blocks: $B = [b_0, b_1, b_2, ..., b_n]$, where each block b_i contains a header H_i and an ordered set of transactions T_i.

Decentralization and Consensus

A blockchain is *decentralized* by nature. However, giving power to the masses gets complicated, especially because agreeing on something is usually hard for a large group. In the late 1970s / early 1980s, research on consensus protocols stemming from

the "Byzantine Generals Problem" [78] laid the groundwork for understanding the challenges of achieving consensus in distributed systems. In his Bitcoin paper [91], Satoshi Nakamoto proposed the first decentralized consensus protocol, known as Nakamoto Consensus (NC) enabling a permissionless set of nodes to achieve consensus on a canonical version of the decentralized ledger, or Blockchain.

Before NC, the development of protocols such as Paxos [77] worked under the assumption of faulty nodes that only crash and do not act arbitrarily. This was later extended by Practical Byzantine Fault Tolerance (PBFT) [38] to tolerate faulty nodes performing arbitrary actions to sabotage the protocol's execution. These protocols, however, work under the assumption of a known set of participants, limiting their scalability and providing an inherent "closed" membership. The assumption of known parties was to avoid *Sybil attacks*, where an attacker can potentially generate multiple identities to gain an unfair advantage in the protocol, e.g., more voting power.

More specifically, this family of protocols is known as Byzantine Fault Tolerant (BFT) protocols, which can also be generalized to *voting-based* protocols. Such protocols can tolerate up to $f \leq N/3$ faulty nodes where f is the number of faulty nodes and N is the total number of nodes in the network. Moreover, these protocols *assumed* that N is a small set of *known* parties and, as a result, do not scale well to a large number of parties.

NC, in contrast, is a *proof-based* protocol that revolutionized consensus by allowing a large, unknown set of participants to achieve consensus without prior acquaintance. This innovation provided a method to counter Sybil attacks and expanded the reach of decentralized consensus. Thus, proof-based protocols like NC represent a significant breakthrough in consensus mechanisms, enabling unprecedented practical agreement among unacquainted parties.

Being an evolving field, definitions for decentralization and consensus are manyfold. This thesis proposes the following informal definitions for the purpose of establishing a common ground:

- **Decentralization**: the property of utilizing multiple nodes N to maintain a replicated ledger L that represents a blockchain, such that instead of a single entity, each node can participate in determining the state of L.
- **Consensus Protocol**: a protocol enabling a large set of N spatially separated entities to order a certain set of values in the system (i.e., transactions modifying the system state) is known as a consensus protocol. In the field of blockchains, various consensus mechanisms (Proof-of-Work (PoW), Proof-of-Stake (PoS), etc.) are employed to agree on valid transactions and their order in a block.

Blockchain Security

To be considered secure, a blockchain and its underlying consensus protocol need to satisfy the following properties:

- **Liveness (L)**: for any valid transaction T_j broadcasted to the network, there exists a probability $P > 0$ that a participant in the network will incorporate T_j into the next block b_{i+1}. In other words, this property states that with high probability, something good eventually happens.

$$P(\text{Transaction } T_j \text{ is included in } b_{i+1}) > \varepsilon, \tag{2.1}$$

  where ε is a small positive value.

- **Safety (S)**: once a transaction T_k is confirmed and included in block b_m at height h, it is computationally infeasible for an adversary to alter T_k or any prior block b_i $(i < m)$. In other words, this property tells us that with high probability, nothing bad ever happens.

$$P(\text{Altering } T_k \text{ or } b_i) < \delta, \tag{2.2}$$

  where δ is some non-negligible quantity.

- **Validity (V)**: a transaction T is considered valid if it satisfies predetermined conditions which, in real world systems, typically include:
  - Correct digital signature verification.
  - Validity of inputs, i.e., the inputs are unspent.
  - Compliance with consensus protocols.

In addition to the properties listed above, properties such as consistency, immutability, and finality reflect more specific characteristics related to the integrity and permanence of a blockchain. We offer the following definitions:

Consistency, Immutability, and Finality:

- **Consistency**: all nodes eventually converge to the same valid blockchain state, maintaining agreement on the ledger's contents and sequence of blocks.

- **Immutability**: the blockchain's cryptographic structure ensures that altering any block is difficult or irrational, i.e., the cost of altering the past blocks is higher than the potential gain from doing so. In terms of Bitcoin, it requires resolving hash puzzles for subsequent blocks, making unauthorized modifications computationally impractical. Ethereum requires owning a supermajority of the total monetary stake in the network.

- **Finality**: finality in a blockchain can take two forms, i.e., eventual or instant. On the one hand, eventual finality states that after some pre-defined time, a transaction cannot be reversed or removed. This follows from the *immutability* and *safety* properties of a blockchain. In PoW blockchains, eventual finality is typically achieved after 6 blocks since the block where the transaction is included. On the other hand, instant finality guarantees that a transaction cannot be reversed or removed as soon as it is added to a blockchain. BFT-based blockchains such as Cosmos[3] guarantee instant finality.

Armed with this understanding of a blockchain and its properties, let us see how Bitcoin satisfies them using cryptographic primitives and network-level assumptions.

The case of Bob, Alice, and Bitcoin

In Bitcoin, the blockchain is maintained by thousands of nodes spatially separated over large distances. These nodes are computers run by various entities worldwide, and they can join and leave the network at will.

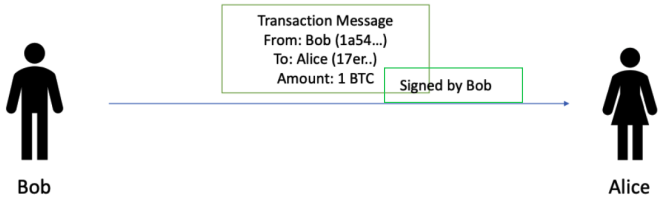


Figure 2.2: Bob pays Alice 1 Bitcoin

As depicted in Fig. 2.2, when Bob pays Alice one Bitcoin, he formulates a *transaction message* that, among other things, specifies from whom this transaction comes and for whom it is intended, along with the amount it transfers. A transaction is also accompanied by a digital signature from Bob, the sender. One might wonder why this signature is needed. The signature is an essential part of this transaction: anyone can forge a transaction from Bob's account without it, so the protocol must have an authentication mechanism to ensure that it was indeed Bob who initiated this transaction. Digital Signatures, specifically EdDSA [67], provide this authentication in Bitcoin.

Bitcoin employs many cryptographic algorithms to provide security in the harsh environment it is deployed in, i.e., a permissionless network where anyone can join or

[3] https://cosmos.network

leave at any time. In this thesis, we explain the necessary primitives only to the degree required for the context in which they are employed. For more information on the cryptographic primitives used in Bitcoin and blockchain technology, interested readers are referred to the thorough systemization of knowledge by Raikwar et al. [102].

Bitcoin utilizes asymmetric cryptography, i.e., a two-key system. Asymmetric cryptography consists of a private key used to authenticate/sign transactions and a public key used to verify a transaction. Along with asymmetric cryptography, Bitcoin also relies on Hash functions.

Digital Signatures: As already explained above, digital signatures are used to provide a means of data authentication as well as non-repudiation[4] of the issuance of a transaction.

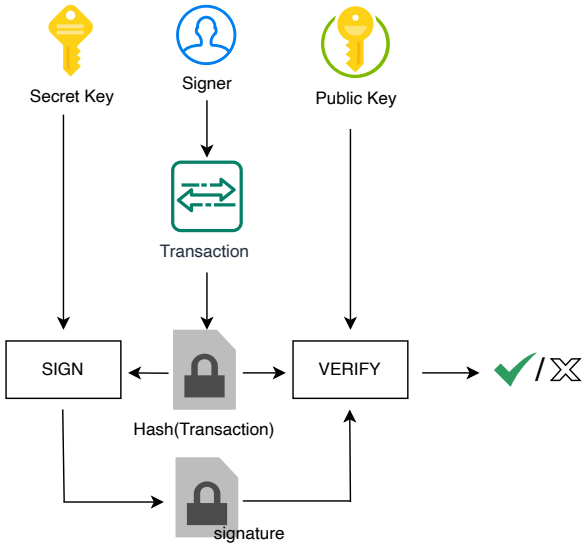


Figure 2.3: Digital signature in Bitcoin

Figure 2.3 shows how users in Bitcoin create and verify a digital signature. The user's private key is used with the *Hash* (a fixed-length digest) of the user's transaction to generate a signature. The verification process involves using the user's public key and signature and verifying whether the signature is generated by the user. Hence, the output is always a Boolean, i.e., a true or a false.

Hash function: As mentioned above, the digital signature is performed on the *Hash* value of a transaction, i.e., $\mathcal{H}(\text{Tx})$. We derive this transaction hash using a primitive known as a cryptographic hash function.

[4]The property of non-repudiation signifies that an action cannot successfully be disputed by its actor.

A cryptographic hash function $\mathcal{H}$ is a powerful tool; it takes an input of arbitrary size and produces an output of a fixed size known as a hash value. Bitcoin heavily leverages the security properties of a hash function [85], such as:

- *preimage resistance*, i.e., for all pre-specified outputs, it is computationally infeasible to find any input that hashes to that output. More specifically, given output x, it is hard to find input y such that $\mathcal{H}(y) = x$.
- *2nd-preimage resistance*, i.e., it is computationally infeasible to find any second input which has the same output as any specified input, i.e., given x, to find a 2nd-preimage $x' \neq x$ such that $\mathcal{H}(x) = \mathcal{H}(x')$.
- *collision resistance*, i.e. it is computationally infeasible to find any two distinct inputs x, x' which hash to the same output, i.e., $\mathcal{H}(x) = \mathcal{H}(x')$.

Bitcoin uses two different kinds of hash algorithms, i.e., SHA-2 [6] and RIPEMD-160 [5]. Note that *collision resistance* is not required for the latter but only the former. Additionally, Bitcoin also utilizes the property of *average search complexity* of hash functions, i.e., it should require on average $2^n/d$ evaluations of $\mathcal{H}$ to find an input x, such that $\mathcal{H}(x) < d$ where $1 \leq d \leq 2^n$, d is the target, n is the number of bits in the hash output, and $\mathcal{H}(x)$ is interpreted as an integer.

Now that the reader is familiar with the main cryptographic building blocks of Bitcoin, explaining what happens to the transaction sent by Bob becomes easy. The transaction sent by Bob is eventually broadcasted to all the nodes in the network. All nodes receive this transaction and verify that it came from Bob (by verifying the transaction's signature using Bob's public key) and that it is valid by checking if Bob has enough money to pay Alice. Given an extensive network of nodes receiving these transactions and subsequently updating their ledgers, not everybody would place the transactions they receive in the same order. This is because some nodes receive transaction A before transaction B, while others might receive them in the opposite order due to network-level delays or malicious activity. Thus, how do we establish a ground truth in this setting?

Nakamoto Consensus

A way to solve this is to let one node among these thousands of nodes publish their ledger version. This leads to an obvious question: how do we choose this node?

Bitcoin employs a novel protocol to answer this question. This protocol randomly selects a node to publish its version of the ledger's state; once it is published, everyone in the network can either agree with this updated state or not. This way, the whole

network agrees on the latest state of the distributed ledger, hence aptly classifying this protocol as a *consensus* protocol. Specifically, Bitcoin's consensus protocol uses PoW to determine the node publishing the next block. This is called NC. It proceeds as follows:

- Nodes collect the latest transactions, verify them, and compile them into a data structure known as a *block*. This block consists of a plethora of information about the recent state transition of the distributed ledger, including the latest transactions that led to this state transition. The information necessary for PoW is:
  1. **Previous block hash** (*prev_block*): the new block generated by each node points to the identifier of the block that the nodes deem to be the latest valid block.
  2. **Hash of all transactions in the block** ($\mathcal{H}(Txs)$): the body of each block consists of a list of transactions, arranged chronologically, leading to the latest state transition. These transactions are also stored in a tree-like data structure (known as the Merkle tree) and hashed in pairs, traversing to the root of this tree. The Merkle tree allows one to quickly check if a transaction is included in a block in time logarithmic in the tree size. This root hash is also an essential component of the PoW protocol.
  3. **Nonce**: This is a random number, or solution, that needs to be found for a node to win and get a chance to publish its block. The PoW algorithm is designed to try different nonce's and brute-force its equation until the desired output is achieved.
- Each node works towards finding the solution, or nonce, to the following equation (which, in Bitcoin, is referred to as 'a puzzle'):

$$\mathcal{H}(\mathcal{H}(Txs), prev_block, nonce) < threshold\,. \tag{2.3}$$

  This threshold is dynamically adjusted to ensure a block is produced on average every 10 minutes.
- The first node to find the solution gets a chance to publish its block and is subsequently labeled as the leader for that round. A round in Bitcoin takes place every 10 minutes and ends when a block is published. The leader is rewarded for their efforts. This reward is in the form of newly minted Bitcoin tokens that were never circulated. Hence, by successfully participating and winning in the PoW protocol, the node introduces (*mines*) new Bitcoin into the system: This is the reason why nodes that participate in Bitcoin's PoW are called *miners*.

NC does not ensure that a single leader is found for a round; there are rounds with multiple leader nodes that each produce valid blocks. If all leaders follow the protocol,

their blocks extend the same *prev_block*, leading to a *fork*. Bitcoin has a mechanism to handle these forks, formally analyzed by Garay et al. in [55].

Hence, the nodes in the Bitcoin network compete to publish their block by following the PoW protocol or NC. This block contains an ordered list of transactions processed in the network since the last block and represents how the state of the Bitcoin network has transitioned since then. A node establishes a ground truth about the Bitcoin network's new state by publishing a block. Once this truth is established, every node in the network verifies whether or not this block is valid based on simple criteria. They verify whether the block points to a valid previous block and that all transactions in the block are:

- unspent;
- accompanied by valid signatures;
- monetary value of their inputs $\geq$ monetary value of their outputs.

Hence, anyone willing to invest the necessary computational resources can become a miner in the Bitcoin blockchain. Anyone can read this blockchain and verify its integrity.

2.2.2 Ethereum

With Bitcoin, the vision of a completely decentralized monetary instrument became a reality; however, is this enough? Bitcoin is limited in its application.

Ethereum [121] is a decentralized platform based on a public blockchain that supports Turing-complete applications. These applications are deployed as SCs which are pieces of immutable code triggered by transactions that utilize the Ethereum Virtual Machine (EVM) for execution. If we had to draw an analogy to describe the difference between Bitcoin and Ethereum, we could envision Bitcoin as a very robust calculator that performs calculations over the most complex equations very securely. At the same time, Ethereum is a computer capable of doing similar calculations while supporting various media playback, gaming, and more. Ethereum's initial version, Frontier, utilized PoW for consensus. However, it drastically differed from Bitcoin in two main aspects:

1. Ethereum utilizes the "account model" instead of the "Unspent Transaction Output (UTXO) model".
2. It supports a general purpose Turing-complete EVM, used to deploy and run distributed code, known as SCs.

Transaction Models

A transaction model describes how a system processes transactions and their resulting changes. This section explains the two transaction models used in Bitcoin and Ethereum: the Unspent Transaction Output (UTXO) and account models, respectively.

- **UTXO Model**: Imagine having ten coins, such that each can be individually spent only once. Once one spends either of these coins, for example, to buy a candy, one now has only nine coins left while the shopkeeper who sold you the candy now has one coin (presuming the candy costs one coin). In this example, each coin can be seen as a UTXO.

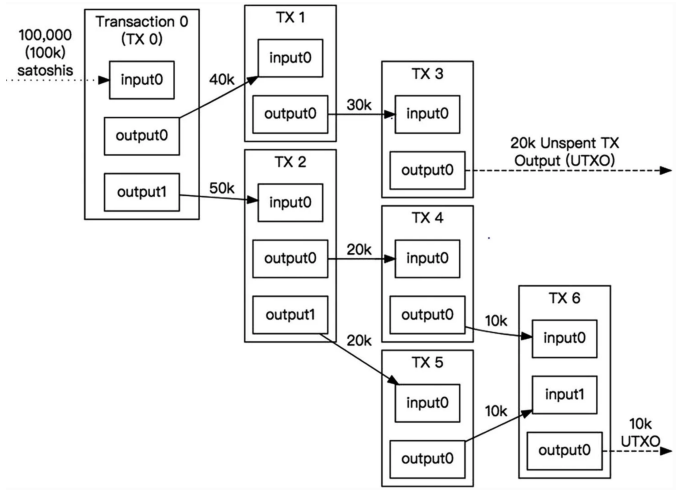


Figure 2.4: Bitcoin transaction input and output

Fig. 2.4 is taken from [56] and shows that each Bitcoin transaction is composed of inputs and outputs, where a transaction takes as input UTXOs and creates new UTXOs as output. For instance, Transaction 0 has as input a UTXO worth 100k Satoshis[5], and two outputs, i.e., output 0 and output 1 for 40k and 50k Satoshis respectively. In addition to inputs and outputs, each transaction in Bitcoin can also have some change (in this case 10k Satoshi), that is returned to the transaction sender as a fresh UTXO. These inputs and outputs could belong to any user who is the sender/recipient of this transaction. Thus, a UTXO constitutes unspent Bitcoins sent to a user by another participant in the Bitcoin network and can be redeemed. Hence, a Bitcoin user's wallet keeps track of a

[5] 1 Bitcoin divided by 10^8 is 1 Satoshi.

list of UTXOs associated with that user's Bitcoin address, and the sum of these UTXOs is the wallet balance.

- **Account Model**: A simple comparison to the modern banking system would make understanding this model straightforward. A bank account's balance is kept in the bank's data servers (centralized storage that every relevant bank employee can access). This storage consists of the account balances of all the bank's clients. In Ethereum, the global state mimics the storage of a bank. It is stored by all participants of the Ethereum blockchain and is known as the *global state*. The global state consists of the balances of each user account (represented as an address) on the system. Figure 2.5 is taken from Vitalik Buterin's introduction to Ethereum [32] and depicts a clear distinction between Externally Owned Accounts (EOAs) that only have a balance and nonce, whereas SC accounts also have an associated code and storage. Each time an Ethereum user makes a transaction, the Ethereum miners/validators check whether the address has enough balance before accepting it as valid. Later, the Ethereum State Transition Function is applied, and the new global state reflects the new balance of the user.

  More specifically, an Ethereum user's account is referred to as a 20-byte address, and it is stored in the global state along with its balance on the Ethereum blockchain. The flow of value and information between subsequent global states is defined by a state transition function, aptly known as the Ethereum State Transition Function. The global state of Ethereum distinguishes between EOAs, controlled by a user's private key, and SC accounts containing executable code, that can be called whenever its address receives a transaction. EOAs are simple user accounts that only store user balance as global variables; however, SC accounts can access the data they store internally and communicate autonomously with other addresses when instructed to.

This difference between Ethereum and Bitcoin is due to the existence of SC accounts that might require keeping track of states in the internal storage to perform various tasks. Bitcoin's UTXO model is stateless; hence, the intuitive approach was to adopt a transaction model to make this easier.

Smart Contracts

Nick Szabo introduced Smart Contracts (SC), which are pieces of immutable code triggered by transactions sent to their address. These SCs live on Ethereum's global state under SC accounts and lie dormant until executed. The Ethereum State Transition Function accommodates the changes due to the execution of a SC.

```
1 pragma solidity >=0.7.0 <0.9.0;
2
```

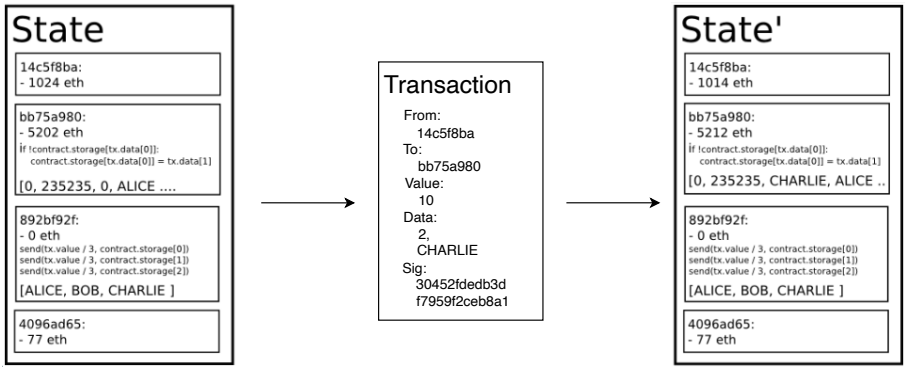


Figure 2.5: The Ethereum State Transition function is applied on each block. The addresses 14c5f8ba & 4096ad65 correspond with EOA, whereas the addresses bb75a980 & 892bf92f correspond with contract accounts

```
3  contract Owner {
4
5      address private owner;
6
7      // event for EVM logging
8      event OwnerSet(address indexed oldOwner, address indexed newOwner);
9
10     // modifier to check if caller is owner
11     modifier isOwner() {
12         require(msg.sender == owner, "Caller is not owner");
13         _;
14     }
15
16
17     constructor() {
18         console.log("Owner contract deployed by:", msg.sender);
19         owner = msg.sender; // 'msg.sender' is sender of current call,
               contract deployer for a constructor
20         emit OwnerSet(address(0), owner);
21     }
22
23
24     function changeOwner(address newOwner) public isOwner {
25         emit OwnerSet(owner, newOwner);
26         owner = newOwner;
27     }
28
29
30     function getOwner() external view returns (address) {
31         return owner;
32     }
```

```
33 }
```

Listing 2.1: A SC written in Solidity to modify ownership [52]

Listing 2.1 depicts a SC written in Solidity. This SC lists the Ethereum address of the entity that deploys it; more importantly, it allows for the change in ownership. Also evident from this listing is that Solidity is a high-level language amongst several in the Ethereum ecosystem. This Solidity code is compiled into bytecode and executed by the EVM.

Borrowing concepts from object-oriented programming, a Solidity SC can be considered as a class that has several functions based on its use-case, can be inherited by other SCs, and, as can be seen in Listing 2.1, also contains a constructor that sets variables upon deployment. Solidity enables SCs to have various programmable features such as modifiers, events, variables, and differing levels of visibility for variables and functions. Interested readers are referred to the official documentation of Solidity[6] for more explanation on these features.

Several domains, such as finance, gaming, sharing economy, storage, etc., have been positively impacted by SCs. They allow for entirely decentralized applications that do not require an intermediary, and the use cases of this technology are theoretically boundless.

EVM: The EVM handles the execution and state transitions of SCs deployed on the Ethereum blockchain. It is built on a low-level stack-based language comprising a set of pre-defined opcodes necessary for maintaining and executing SCs. On a high level, EVM can be viewed as a global computer providing a decentralized platform for all SCs. However, all transactions required for running SCs are locally and synchronously executed on each node in the network. All instructions executed by the EVM have a monetary cost to compensate the nodes executing them. The user bears these costs as '*gas*'. This economic cost also has a security-related facet, i.e., it prevents Denial-of-Service (DoS) attacks as repeated computations prove expensive for the attacker.

Blockchains such as Bitcoin and Ethereum symbolize the power, fairness, and transparency of decentralization. However, at what cost?

[6] https://docs.soliditylang.org/en/v0.8.21/

Chapter 3

The Scalability Trilemma

This chapter starts by discussing how blockchains can be categorized based on their design and membership criteria. Next, we examine the Scalability Trilemma, which claims it is hard to simultaneously achieve scalability, security, and decentralization. We discuss the reasoning behind the Trilemma and the various approaches to address it. Finally, we examine how this thesis explores the Trilemma in multiple domains that use blockchain technology.

3.1 Categorizing Blockchains

Blockchains can be classified into two categories based on their design: monolithic and modular. A blockchain typically consists of three core abstractions:

Abstractions

1. Consensus: abstraction that is used to define the current state of the blockchain that the participants accept;
2. Data availability: abstraction that enables blockchain participants to access the necessary data to reach consensus on the state of the blockchain;
3. Execution: abstraction that enables new transactions to be executed by the participants to update the current state of the blockchain.

Monolithic blockchains are defined as blockchains that carry out these core abstractions in the same layer (namely *on-chain*), i.e., layer one or L1. Bitcoin is a famous example of a monolithic blockchain; all validators maintaining Bitcoin execute $\sim$ 1.5 MB of transactions every 10 minutes (on average) to reach consensus on the latest state.

On the other hand, modular blockchains split these abstractions between different layers; transactions are executed *off-chain*, and the L1 is used for storing and verifying the updated state.

Membership

Additionally, blockchains can further be classified based on their membership constraints, as follows [101]:

- Private blockchain: these blockchains are typically owned by a single entity granting permissions to read/write to participants of their choice. At any point, the blockchain owner can modify/delete data written by the permitted members.
- Consortium blockchains: these blockchains allow a pre-determined set of members to write to the blockchain. However, they can choose to let *anyone* read data. A typical way to obtain membership in such blockchains is identity verification. Hence, every member knows the identity of other members of the blockchain.
- Public blockchains: *anyone* can read and write from/to the blockchain. The constraint to participate in public blockchains is the same for everyone.

The Trilemma

Bitcoin exhibits an inherent problem of monolithic public blockchains — they do not scale well. An optimal decentralized system comprises a combination of three properties, explained as follows:

1. Scalability: a decentralized system should be capable of efficiently operating independently of the number of users;
2. Security: a decentralized system should be secure against any attacks that affect its proper and honest functioning;
3. Decentralization: a decentralized system should achieve scalability and security without needing to depend on intermediaries, hence, without sacrificing its decentralized nature.

The blockchain trilemma, or *Scalability Trilemma*, as coined by Vitalik Buterin [93], states that it is challenging to achieve scalability, decentralization, and security simultaneously. As mentioned before, in the case of Bitcoin, its inherent decentralization and security come at the cost of scalability, as explained below. Alternatively, permissioned and private blockchain sacrifice decentralization for better scalability and security. Arguably, private blockchains further sacrifice security for higher scalability. This thesis focuses solely on public blockchains.

A blockchain's scalability can be defined via two metrics, i.e., the number of transactions it can process per second (*transaction throughput*) and the time it takes for a transaction to be finalized (*transaction confirmation latency*). Bitcoin supports a theoretical maximum throughput of only 27 transactions per second (tps) [57], appearing minuscule when compared to Visa's 47,000 tps [114] and Alipay's 256,000 tps[1]. Bitcoin's one-hour latency makes its mass adoption as a payment method difficult.

3.2 Approaches to Addressing the Trilemma

Bitcoin's design enforces the production of one block every 10 minutes. Initially, the size of this block was set to be one Megabyte (MB) until the size parameter was changed to accommodate a maximum of four million "weight units" in 2017. Consequently, Bitcoin's block size can theoretically be as large as four MB. Therefore, Bitcoin (by design) is constrained to support only a limited number of transactions per block. This combination of limited space and high latency is Bitcoin's most significant scalability inhibitor.

Let's try first to address a simple question: Why can't we change Bitcoin's design to address its scalability issues? Supposedly, we could either increase the block size to accommodate more transactions or decrease the block latency.

Increasing the block size

Theoretically, a larger block would provide more space for transactions to be included within the block's latency period, thus enabling more transactions to be processed within each block. This would consequently help Bitcoin sustain a more extensive user base efficiently. However, in practice, a larger block size would affect public verifiability. Due to the increased storage and bandwidth requirements, the membership constraints for joining the Bitcoin network to maintain it would become harsher.

[1] https://expandedramblings.com/index.php/alipay-statistics/

Reducing block latency

Decreasing the block latency would have a similar effect on helping Bitcoin efficiently sustain a more extensive user base. However, the network's security would suffer an adverse effect due to the more frequent formation of *forks*, increasing the uncertainty in reaching consensus. Forks are a harmful byproduct of the leader election mechanism in blockchains that may elect multiple leaders for the same round. These leaders then produce alternate versions of the blockchain, from which the canonical version has to be selected. In Bitcoin, this selection is done by following the "longest-chain" rule, which states that in the presence of alternate chains (forks), the longest one should *always* be selected as it denotes more *work*. A result of the longest-chain rule is the requirement of a transaction to wait for six blocks (after it has been included in a block) to be finalized.[2] This is because the chain that includes a block with that transaction and six more blocks is, with a very high probability, the longest. Thus, it will be selected as the canonical version.

Additionally, whenever nodes in the network do not have some transactions included in propagated blocks (known as fresh transactions), extra communication is required to synchronize them. This mechanism increases the block latency, i.e., if the block interval is reduced, there will be more fresh transactions, which increases the block latency and decreases the transaction throughput [129].

Thus, statically changing Bitcoin's base design would improve scalability while adversely affecting decentralization and security. As predicted, this would correlate directly to the Scalability Trilemma. It may be possible to measure the evolution of the Bitcoin network's network and storage capacity to *dynamically* adjust the block size. However, this approach may not provide strong security [128] and would require a *hard fork*[3]. Hence, its benefits and efficacy require further research.

However, addressing scalability issues in public monolithic blockchains is not a lost cause, as other techniques to address this issue exist. A discussion on these techniques is given below.

Tweaking the consensus protocol

PoW consensus, introduced in Bitcoin, is a secure and robust consensus protocol enabling untrusting nodes to arrive at an agreement about the state of a protocol at a given time in a fully decentralized fashion. This feat, however, comes at the cost of efficiency. Proving a hard computation requires time (by design) and a lot

[2] Once a transaction is deemed final, it cannot be mutated or reversed, and is guaranteed to be a part of the canonical version of the blockchain. This property is known as Transaction Finality, and in Bitcoin, this can take up to an hour on average.

[3] A radical change to the protocol of a blockchain network

of energy. This inefficiency, coupled with the already inflexible design of public monolithic blockchains, makes PoW infamous for scalability. Other alternatives to PoW, such as PoS, Proof-of-Capacity (PoC), Proof-of-Space-Time (PoST), Proof-of-Burn (PoB), Proof-of-Authority (PoA), Proof-of-Elapsed-Time (PoET), etc., have surfaced [23], promising better efficiency than PoW. Out of these alternatives, PoS is arguably the most popular; blockchains such as Ethereum, Cardano[4], Solana[5], and Mina[6] all employ variants of PoS consensus, consequently offering a significantly better throughput. However, this comes at the cost of reduced security [104, 94], re-enforcing the Scalability Trilemma. Additionally, by transitioning to PoS, Ethereum's transaction throughput has improved; however, its transaction finality now takes ∼15 minutes [34]. This drastic increase in Ethereum's finality time is a design decision, where in order to be *finalized*, each block needs to be attested by validators who control more than two-thirds of the total stake in Ethereum in two epochs (time-periods) that last for 6.4 minutes each.

Sharding

Blockchains verify transactions by re-executing them. This re-execution makes the process of transaction verification inefficient, especially since it has to be done by all maintainers (miners in PoW and validators in PoS) of the blockchain. Sharding splits the set of maintainers into subsets (*shards*) such that each shard performs transaction validation in parallel. This improves the efficiency of transaction validation by splitting the verification load across the whole network. The Ethereum 2.0 roadmap is progressively transforming the entire blockchain to become modular and using sharding as one of its core functionalities to scale its performance massively. However, sharding is a very complicated primitive, especially when a transaction has to be validated across multiple shards. Such transactions are relatively common and utilize transactions that belong to numerous other shards as inputs. Furthermore, building adversarial transactions that would clog the network is fairly easy.

Off-chain solutions

These solutions substantially offload data processing from the blockchain by performing it on a separate interface (layer). A wide variety of such solutions is utilized in practice. Amongst the most popular are payment channels, side chains, commit chains, and rollups. A major distinction between these solutions is their utilization of the underlying blockchains. Layer-2 solutions such as channels and rollups rely on the underlying

[4] https://cardano.org
[5] https://solana.com
[6] https://minaprotocol.com

blockchain for security and dispute resolution, whereas side chains communicate with the underlying blockchain to receive inputs and deliver outputs. These solutions are depicted in Fig. 3.1 and Fig. 3.2 and described as follows:

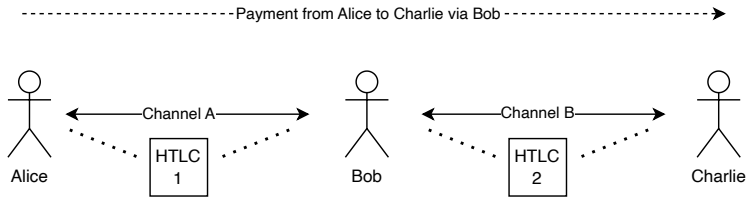


Figure 3.1: A payment channel network routing a payment between Alice and Charlie via Bob and ensuring atomic payments by using Hashed Time Lock Contracts (HTLC)

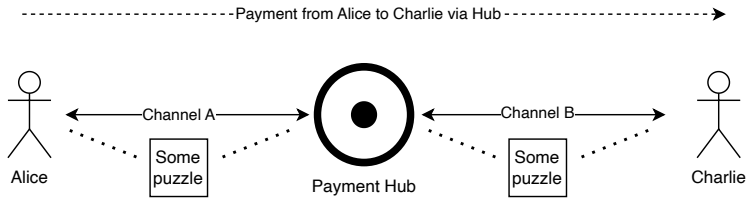


Figure 3.2: A payment channel hub routing a payment between Alice and Charlie and ensuring atomic payments by using an arbitrary puzzle (e.g., RSA) puzzles)

- **Channels**: blockchain channels can be divided into two categories, i.e., payment- and state-channels. Both variants constitute three phases: (i) opening, (ii) off-chain modification, and (iii) closing. Payment channels [47] are specifically designed to scale payments and find applications in scenarios requiring micropayments. A generalization of this approach that considers general-purpose computation (beyond payments) is encapsulated by state channels. Both payment and state channels require locking up funds, known as the collateral. However, state channels also require knowledge of the current state of the application for which they are built. The 'off-chain modification' phase allows for a theoretically unbounded number of transactions, helping blockchains scale to many users without incurring additional computation load. Channels are typically established between two users and can be extended in the following ways:
  - Channel Networks: as depicted in Fig. 3.1, such solutions utilize established channels between parties to route payments. For instance, if Alice and Bob have an established channel (channel A), and Bob and Charlie have another established channel (channel B), channel networks utilize Bob

as an intermediary for transferring money between Alice and Charlie and ensure atomic payment by utilizing Hashed Time Lock Contracts (HTLCs). A popular example of a payment channel network is the Lightning Network [99] deployed on Bitcoin;

  - Channel Hubs: as depicted in Fig. 3.2, these solutions utilize a payment hub to route payments between parties. A payment hub is an intermediary with many channels such that the participants maintaining a channel with the hub can route payments between each other while using the hub as an intermediary. A famous example of a payment hub is TumbleBit [63].

- **Rollups**: these solutions leverage a combination of off-chain computation and on-chain verification. Rollups come in two flavors: optimistic- and ZK-rollups [33]. Each of these variants is described as follows:

  1. Optimistic rollups: aptly named, optimistic rollups operate on the principle of optimistic execution. Transactions are handled on an auxiliary layer known as the "rollup chain," where they are validated and executed, leading to a state transition. The execution summary on the rollup chain is then submitted on-chain, where it is verified for any 'frauds'. If a problem with this execution summary is found, it can be disputed with a 'fraud-proof' and resolved on-chain.
  2. ZK-rollups: by relying on advanced cryptography such as Zero-Knowledge Proofs (ZKPs), these rollups provide inherent security and trustlessness while improving the underlying chain's scalability. Similar to optimistic rollups, all transactions are executed on the rollup chain, yet a single ZKP proves the correct execution of all transactions in the rollup batch. The proof and some 'call data' necessary to verify it is submitted on-chain, where a smart contract verifies the correct execution and formulation of the new state. ZK-rollups are significantly faster in terms of verifiability when compared to optimistic rollups, with time for verification of ZK-rollups taking a few seconds as opposed to optimistic rollups taking weeks. However, unlike optimistic rollups, ZK-rollups do not offer EVM support by default.

  Rollups significantly increase a system's transaction throughput at the cost of increased computation for rollup operators and do not improve the transaction confirmation latency.

- **Sidechains**: these solutions enhance scalability by operating on independent parallel chains using their operators and security assumptions while still linked to the main chain. Sidechains usually leverage a more flexible and efficient execution mechanism than the main chain; hence, they help alleviate congestion on the main chain by offloading some of its execution load.

The types of sidechains can vary significantly based on the kind of blockchain application, including considerations such as security, decentralization, trust, and use case compatibility.

The Scalability Trilemma remains an issue attracting considerable research efforts. From changes in the blockchain design to sharding and finally to off-chain solutions, solutions to the Trilemma show their advantages in theory; however, none has yet ideally addressed the Trilemma in practice.

3.3 The Trilemma in Practice[7]

This section discusses how this Trilemma holds up in practice, specifically in use cases motivated by the sharing economy and energy domain.

3.3.1 Sharing Economy

Sc2Share: Our paper, called Sc2Share [83], is an attempt to re-invent the car-sharing economy, and showed Ethereum's inefficiency in practice. Sc2Share's main goal is to provide a decentralized alternative for bookings and settlements, supporting the rapidly growing car-sharing market. As of 2018, all practical solutions supporting the sharing economy rely on centralized platforms for exchanging information and settling payments. Unfortunately, a series of incidents indicates that these platforms are not as trustworthy as they promise. Their centralized membership system and pricing scheme allow them to undermine *fairness*. Uber (a well-known peer-to-peer car-sharing platform) charges its customers based on an algorithmic prediction of how much they are willing to pay rather than the services rendered [95]. Furthermore, Airbnb, an online marketplace for sharing accommodation, has been in the news for targeting underrepresented populations to exclude them from using its services [41]. This example highlights that there is no mechanism to prevent these services from misusing their power, such as targeting a minority group.

Given the worldwide growth in car-sharing service users between 2012 and 2014 by 170% [112], the need for a more robust and secure car-sharing system seemed of the utmost importance. Our goal was to provide a decentralized platform enabling billings and settlements for the use case of sharing privately owned cars. By relying on the Ethereum blockchain, we satisfied several security properties and developed a fully functioning decentralized application capable of hosting car-sharing advertisements, executing billings and settlements, handling disputes, etc.

[7] This section reuses some text from our publications [83, 46, 89, 31, 84, 82, 81]

However, Sc2Share had long waiting times for each operation. The confirmation latency of each transaction on Ethereum used to be ~ 3 minutes on average; this latency also depends on other factors, such as the fee included in the transaction. This is due to the rational behavior of blockchain miners to maximize their revenue. This confirmation latency had increased almost four times over when Ethereum transitioned from PoW to PoS and is now ~ 13 minutes per transaction on average [53].

Note that this confirmation latency time captures the time necessary to ascertain absolute confidence that a user's transaction would be included in the canonical blockchain. Users can also instantly accept a transaction as soon as the transaction appears on the blockchain, similar to how Bitcoin Automated Teller Machines (ATMs) work. However, it has been shown both in theory [68] and practice [100] that doing so is dangerous. The risk of a successful double-spending attack is highest when a new transaction is appended to the blockchain via a new block and exponentially decreases when successive blocks are added on top of the block containing it; the convention has been to wait for six blocks or *confirmations* in PoW blockchains. With Sc2Share, we realized that PoW Ethereum, if used alone, was not ready to support sharing economy use cases requiring fast transaction finality and a large number of users. As depicted in Fig. 3.3, our work lies at the bottom end of the Trilemma, signifying its secure and decentralized nature and its shortcomings in terms of scalability.

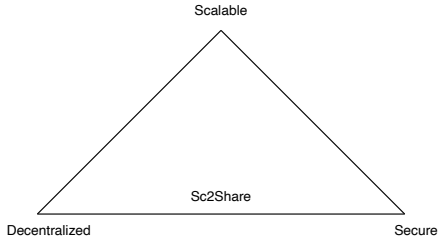


Figure 3.3: Sc2Share in the Scalability Trilemma

Following the results we obtained from Sc2Share, we became curious about existing decentralized applications (dApps) in the sharing economy domain and how they fit into the Scalability Trilemma. Storage was proposed as another field that could be revolutionized by Ethereum-like blockchains. Hence, the domain of decentralized storage attracted our interest.

Storj analysis: Nowadays, behemoths such as Amazon, Google, and Microsoft abstract away the complexity of storage and store our data in large data centers (i.e., clouds).

As mentioned earlier, the centralized architecture these client-server cloud providers use introduces many problems. Firstly, because a single point of failure/attack exists,

it makes them more vulnerable to data breaches [96, 58], data outages [15, 116] and facilitates censorship [51]. Secondly, a lack of privacy and data ownership allows big corporations to misuse our data in a way that best befits their for-profit strategy [130, 35]. Thirdly, as the cloud storage market is not nearly as competitive as it could be, prices can be kept artificially high [109, 113, 108].

Numerous initiatives have been undertaken to develop (partly) decentralized storage solutions that no longer suffer from these problems. This, in turn, has given rise to open-source projects such as Sia [118], Storj [76], Swarm [111], Filecoin [54], and many others. Compared to traditional client-server cloud providers, these systems are characterized by peer-to-peer (P2P) architectures that significantly lower infrastructural requirements. This is one of the main reasons these projects can provide the same (or even higher) data availability and durability levels at lower prices. Most projects also guarantee data ownership and privacy by encrypting any data before it leaves the customer's device.

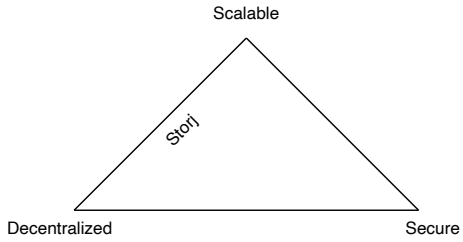


Figure 3.4: Storj in the Scalability Trilemma

In our paper "Exploring the Storj Network: a Security Analysis" [46], along with providing a detailed architectural overview of Storj, we also show its vulnerability to a DoS attack. Our attack was performed in their dev/.test environment and made an important system component unavailable, leading to potential data and monetary loss. To wit, Storj's architecture requires interaction between three major components, namely: (i) **Storage Node** that is responsible for leasing their excess hard drive space to customers in exchange for monetary incentives, (ii) An **Uplink** that enables interaction between a customer and the Storj system, and (iii) A **Satellite** that manages the whole system and performs duties that include storing metadata, paying storage nodes and executing audits.

Storj's decentralized architecture resembles a stratified topology, where the satellites uniquely maintain the network and incentivize peers. Hence, the Satellite's failure would negatively affect the system as the users could not access their data. Thus, the Satellite became our prime target for a DoS attack. After successfully adapting Storj's client-side code, we forced a storage node to overwhelm a satellite with recursive

requests to establish a connection, eventually causing the satellite to crash. This crash not only leads to service downtime but also to potential loss of data.

Interestingly, Storj achieves high scalability since it operates in a modular fashion. The data storage and communication between nodes happen off-chain, whereas the payments happen by utilizing the Storj token on the Ethereum blockchain. As shown in Fig. 3.4, Storj's position in the Trilemma signifies its advantages, such as decentralization and scalability over *lower*[8] security.

3.3.2 Energy Domain[9]

Part of the research done in this thesis was done as part of an FWO SBO project called Secure and Privacy-Friendly Peer-to-Peer Energy Trading (SNIPPET). This research involved investigating the various subdomains of the energy domain (in addition to P2P energy trading) and understanding the impact of decentralization, specifically blockchain technology, on them. Numerous academic and industrial research projects in the energy domain suggested using blockchain technology for many use cases. Yet, most of them built their solutions using permissioned blockchains due to the scalability issues present in permissionless blockchains. The use cases that comprise this domain include but are not limited to, wholesale energy trading, P2P energy trading, green energy certificate trading, electricity consumption optimization (demand management), Electric Vehicle (EV) charging, and tariff decisions (load-balancing).

The use cases mentioned above are described as follows:

1. **Wholesale Energy Trading**: current energy trading is carried out in three different markets: wholesale, balancing, and retail. Electricity suppliers primarily use wholesale markets to purchase electricity from generators (GENs). Wholesale markets consist of several inefficient procedures due to the sheer complexity of how these markets are designed, and each transaction needs to be repeatedly verified from initialization to final settlement [18]. The additional costs arising from this inefficiency make participation unfavorable for small-scale participants. The main idea for using blockchain technology in the wholesale energy trading use case is to remove intermediaries leading to inefficiency, making these markets more inclusive for minor participants. Prominent examples of applying blockchain technology to energy trading are EnerChain [88] and Grid+ [42].

2. **P2P Energy Trading**: these markets represent an inherently decentralized form of energy market, where prosumers (energy-producing customers) and customers

[8] Our experiments only show the DoS vulnerability in Storj's dev/.test environment.
[9] This subsection reuses some text from our publications [31, 89, 84]

directly trade their generated surplus, altogether avoiding any intermediaries. Hence, by nature, blockchain-based systems seem to be apt to this use case. Many academic research papers [86, 105, 87] suggest solutions for P2P energy trading by using Distributed Ledger Technologies (DLT).

3. **Green Energy Certificate Trading**: tradable commodities that represent an amount of green electricity generated can be traded among participants in a trading market. These certificates are helpful in many scenarios, such as compliance markets where suppliers need to prove that they have a certain percentage of renewable production in their portfolio. Current market structures that enable such a use case are complicated [18]. Hence, green energy certificate trading has captured the attention of experts trying to simplify such markets using blockchain technology. A prime example of gaining interest in combining green energy certificate trading and blockchain technology is the pilot launched by Nasdaq [37]. Academically, research performed by Castellanos et al. [36] uses a private instance of Ethereum, and a public blockchain is suggested by Knirsch et al. [70] to perform energy certificate trading.

4. **Electricity Consumption Optimization**: this use case requires participants of local energy communities to submit strategies that optimize the demand and supply of electricity. The goal is to be as independent as possible from the main grid. This is achieved by reducing the amount of electricity (1) fed to the grid, and (2) withdrawn from the grid. The application of blockchains typically increases the transparency of various strategies to all participants in the system, which in turn allows for more efficient strategies. Research by Knirsch et al. [71] suggests using a Directed Acyclic Graph (DAG) for this use case; additionally, Munsing et al. [90] proposes the use of blockchain technology to enable a community-driven optimization process.

5. **EV charging**: modern transport by nature is decentralized, with ride-sharing services such as BlaBlaCar[10] and on-demand transportation services such as Uber on the rise. The vast web of passengers, vehicles, drivers, and charging stations calls for robust and decentralized services for simplicity. Knirch et al. [73] propose a blockchain-based protocol to enable a decentralized EV charging interface while keeping the actual choice of blockchain agnostic to their protocol. Su et al. [110] propose a permissioned blockchain-based architecture enabling decentralized EV charging.

6. **Tariff Decisions**: the emergence of smart grids offers opportunities for smarter tariff decisions, i.e., customers may favor cheaper energy tariffs and schedule their energy consumption accordingly. Such possibilities open up venues for modular tariffs and dynamic pricing that incorporate the overall grid capacity and customer-centric demands [72].

[10] https://www.blablacar.com

Following the exploration of use cases in the energy domain, we understood that most proposed solutions employ private or permissioned blockchains [31]. The ones that suggest using public blockchains, e.g., [73], leave their solution's scalability questions for future work. Hence, the usability of a public blockchain in the context of these use cases remained questionable; due to the typical "high workload" of transactions in the case of energy domain use cases, an analysis of whether scaling techniques native to public blockchains can be used in this context was crucial.

As discussed in Sect. 3.2, state channels enable offloading general computations from the main chain to an overlay network. Hypothetically, utilizing state channels in the energy domain would disregard the possibility of the main chain serving as a performance bottleneck.

Hence, in our work titled "Off-chain state channels in the energy domain" [31], we devised a methodology to classify whether state channels are applicable in these use cases. Our methodology was inspired by the famous "Do you need a blockchain?" paper by Wüst and Gervais [124]. Our proposed methodology is detailed on page 123 and consists of the following questions:

- **Does the use case require using a blockchain?**

  We utilized the methodology proposed by Wüst and Gervais to answer this question.

- **Does the use case involve the exchange of assets?**

  Transfer of assets lies at the base of channels; in payment channels, this transfer involves exchanging monetary value, and in state channels, it involves any arbitrary asset. Hence, if a use case does not include asset exchange, we argue that utilizing state channels would not provide any benefits.

- **Does the use case involve a monetary deposit?**

  Atomicity in channels results from monetary deposits that serve as collateral in case of misbehavior; more often, they are the source of monetary remuneration. If a use case does not require a financial deposit, we argue it does not need to support state channels.

- **Does the use case benefit from instant finality?**

  As mentioned in Sect. 3.1, the latency of transaction confirmation is a metric to measure the scalability of a blockchain system. State channels provide instant finality to applications, i.e., the transactions are confirmed inside the state channels within a few milliseconds. Hence, a natural question is whether a use case in the energy domain benefits from this property.

- **How many transactions does the use case incur on the main chain?**

> State channels require one transaction to be instantiated and a second transaction to be finalized. Hence, each use case that utilizes state channels incurs at least two main chain transactions. Thus, if a use case has a relatively low transaction load on the main chain, we argue it does not benefit from state channels.

By applying this methodology to the above use cases, we concluded that state channels are most beneficial for wholesale energy trading, EV charging, and energy consumption optimization. Since other use cases do not benefit from state channels, deploying them on a public blockchain would always lead to a performance bottleneck. Consequently, the most beneficial is the utilization of permissioned or private blockchains, as suggested in the original architectures of these researches.

A principal research question during the SNIPPET project was: "Can blockchains be utilized to perform billings and settlements in the context of local energy trading?". However, as concluded in our aforementioned paper [31], local energy trading does not benefit from using state channels. Local energy trading involves ephemeral relationships between prosumers and consumers based on an auction mechanism that matches bids and offers to trade electricity. Hence, the number of transactions needed to set up a state channel might already be more than the transactions required to trade electricity between a prosumer and consumer, given that they only trade once.

However, a stratified approach similar to how Storj works could be considered a viable option to carry out local energy trading, where the users participate in an off-chain trading platform to match their bids and offers and perform settlements of bills that are derived based on their production and consumption of electricity orthogonally; e.g., by utilizing a blockchain-based cryptocurrency. Hence, we hypothesized that combining local energy trading and blockchains would still lead to a decentralized platform where all procedures required in local energy trading, i.e., market making, billings, and settlements, could be carried out in a privacy-preserving and decentralized fashion.

As a first step towards realizing this goal, in our paper titled "Sharing Economy in Future Peer-to-Peer Electricity Trading Markets: Security and Privacy Analysis" [89], we formulated the different forms that the future local energy trading markets could take and the security and privacy requirements these markets would involve. Our analysis resulted in the following types of scenarios:

1. Direct & active peers (**S1**): in this scenario, prosumers and consumers would actively participate in the market and trade directly to maximize the extractable value from the local energy markets.

2. Direct & passive peers (**S2**): in this scenario, prosumers and consumers would not actively participate in the local energy markets but instead utilize representatives

that actively participate and trade with other representatives to maximize their client's extractable value from the local energy market.

3. Indirect & passive peers (**S3**): in this scenario, prosumers and consumers utilize representatives similar to the previous scenario; however, the representatives do not trade directly with each other but utilize an intermediate trading platform that settles all trades.

4. Indirect & active peers (**S4**): in this scenario, prosumers and consumers would actively participate in trade; however, they would not do this directly and instead involve an intermediate trading platform that settles all trades.

Having covered the various scenarios in our paper, we analyzed how these scenarios could have potential security and privacy shortcomings. We discuss them as follows:

- 'S1' is a completely P2P model for trading electricity and is potentially vulnerable to network-level attacks such as Sybil attack or DoS, i.e., an attack where the adversary renders a service inaccessible to its users by causing a server shutdown or network downtime; hence, an implicit requirement for this scenario to succeed is secure authentication and secure congestion policing feedback [80].

- 'S2' involves a stratified topology where representatives are essential in coordinating the trading for their clients, i.e., prosumers or consumers. Hence, they become single points of failure for attacks that might reveal their clients' personal information and even prevent them from executing trades. In addition to secure congestion policing feedback techniques, they also require distributed and robust storage mechanisms such as IPFS [24].

- 'S3' also symbolizes a stratified topology with the addition of a 'broker' or trading platform settling the trades executed by the representatives for their clients. 'S3' opens the system's availability risks due to the reliance on settlements delegated to the broker. This is due to the broker acting as a single point of failure; hence, an implicit requirement for a broker-based scenario to succeed is the distribution of this role between several parties, e.g., by using Multi-Party Computation (MPC) [79].

- Finally, 'S4' creates privacy issues, such as the broker being able to infer information about specific prosumers or consumers because the broker has access to all bids and offers in the local energy trading market; thus, advanced cryptographic techniques that allow computing over encrypted data such as homomorphic encryption [39] are required in this scenario.

In subsequent research during the SNIPPET project, our colleagues proposed privacy-preserving mechanisms for demand forecasting in the context of bids and offers in

local energy markets [97]. They also devised a privacy-preserving auction mechanism that utilizes MPC and does not reveal information about any participating prosumer or consumer [44]. The auction mechanism was executed by a distributed trading platform (or a distributed broker), and as we predicted in [89], utilized MPC for avoiding a single point of failure. Additionally, they also utilized computing over encrypted data techniques to preserve the privacy of prosumers and consumers. This approach symbolized 'S4' in practice and still required a way to carry out billings and settlements.

Such auction mechanisms require the users to submit bids/offers to buy/sell electricity for future trading periods. However, the committed volumes usual differ from the volumes consumed/injected. This happens because of the following factors: (1) volatility of the consumption/production patterns of the households [74], (2) uncertainty in the outputs of local renewable sources (RES) [62], and (3) the inaccuracy of prediction algorithms used by households to devise their auction bids [119]. These discrepancies in electricity volumes not only disrupt the operation of the grid but also lead to an increased cost of balancing the grid [50]. This cost may possibly be passed on to the users, hence, has to be minimized as much as possible.

In our paper titled "Billing Models for Peer-to-Peer Electricity Trading Markets with Imperfect Bid-Offer Fulfillment" [84], we devised several billing models that considered imperfect bid-offer fulfillment and maximized extractable value for prosumers and consumers. Our billing models took the individual deviations of P2P market participants and split the cost incurred amongst these participants in four ways: (i) Individually, where each participant was responsible for their own deviation; (ii) Socially, where deviation of prosumers and consumers are aggregated respectively and socially split amongst all deviating prosumers or consumers; (iii) Universally, where all individual deviations are aggregated into a total deviation and shared amongst all deviating P2P participants, regardless of them being prosumers or consumers; (iv) Universally with a new *mid-market*, where non-P2P participants are allowed to buy the deviations at a better price than retail.

The billing models we proposed in [84] were not designed to be privacy-preserving. Naturally, the billings and settlements mechanism we devised had to strictly adhere to the same privacy guarantees of the other solutions addressing different aspects of P2P trading (such as demand forecasting and auctions) in the SNIPPET project.

Hence, in our paper "Fully Privacy-Preserving Billing Models for Peer-to-Peer Electricity Trading Markets" we proposed and implemented a fully privacy-preserving billing model with a *universal cost-split*. To wit, we utilized Fully-Homomorphic Encryption (FHE) and modified the algorithm of our billing model with universal cost-split (as proposed in [84]) to reveal no personal information of the households. We ensured that (i) the private consumption/production data of all households is never made public and (ii) the prosumers and consumers are never linked to each other while guaranteeing the integrity of calculated bills. Some background on FHE is given on

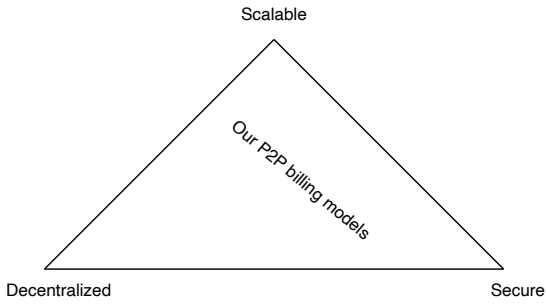


Figure 3.5: Our billing models in the Scalability Trilemma

page 160 of this thesis.

As depicted in Fig. 3.5, our fully privacy-preserving billing model lies in a "sweet spot" in the Scalability Trilemma, i.e., despite being decentralized it trades off availability for efficiency. This is because of our utilization of a stratified topology, where the grid operator is responsible for running the billing model. Despite this concentration of responsibility, any data generated by the households is never revealed to a third party, regardless of them being a part of the billing protocol or not. Hence, our billing models provide full data ownership and efficiency, while trading off availability. Thus, we position our work more towards *scalability* and *security* in Fig. 3.5.

Our billing models consider the following entities:

- Prosumers and Consumers: The entities who pay the bills derived by the proposed billing models are called prosumers and consumers. Prosumers are participants in the P2P energy trading markets that produce and consume electricity; whereas, consumers only do the latter.
- Smart Meters (SM) and Home Energy Management System (HEMS): These inbuilt devices provide real-time measurements of production/usage of the households they are installed in. Additionally, they also aid in facilitating P2P trading by performing functions and providing information necessary for our proposed billing models to work.
- Suppliers: In all our proposed billing models, the suppliers act as a fall-back options in case of any deviation caused by the P2P energy trading market. Trading with suppliers usually incurs a Retail Price (RP) that is higher than the Trading Price (TP) of P2P energy trading and is considered a penalty for the deviating users.

- Grid operator: The entities that manage the injection and output of electricity to/from the grid are known as grid operators. They need to ensure that the grid is always balanced in order to avoid any outages. In our system, we assume that grid operators also perform aggregation of inputs from households to calculate the total deviation in the system and run the billing models.
- Market Operator: The market operator takes the privacy-preserving offers/bids from the prosumers/consumers and runs a blind auction mechanism to match them and determine a TP. Once its function is complete, only the participating prosumers and consumers know of their eligibility to participate in the P2P trading as this is done in a privacy-preserving way [14].

In addition to being privacy-preserving and fair, our billing models are decentralized and utilize the stratified topology with the grid operator playing the unique role of an *Aggregator*. The Aggregator entity is responsible for running the *server* side of the billing model algorithm and receives inputs from the *client* side, i.e., the households. For each period, the Aggregator is also aware of the total deviation incurred by the P2P market concerning the promised bids and offers. Using this total deviation that is accessible to everyone, along with the household-specific encrypted consumption/production data and their encrypted individual deviations, the Aggregator computes a bill for each household in every period. No entity in the entire P2P market has complete information about the market, and inputs from all parties are required for our billing models to work.

The main contribution of our billing models (along with being privacy-preserving and decentralized) is that they consider possible deviations incurred due to the demand forecasting of the promised bids and offers provided by the households without revealing any sensitive information to third-parties, even at the time of bill calculation. In addition, the calculated bills *remain* encrypted, unless decrypted by the households themselves; to the best of our knowledge, our work is the first to do so.

3.3.3 Financial Domain[11]

Settlements in our billing models are concluded with the *supplier*. Following our modular approach, the settlements could either be done using a designated fiat currency based on the market the household is based in or using any cryptocurrency of a user's choice, assuming that their electricity supplier supports payments using cryptocurrencies. However, would it be realistic even if their supplier would support cryptocurrency payments?

[11] This subsection reuses some text from our publications [81, 82]

As established before, significant cryptocurrencies such as Bitcoin and Ethereum incur a high transaction confirmation latency. Even if this high latency is suitable in the context of settling bills, due to the sheer number of transactions executed on the blockchain for settling bills, the transaction fees will increase considerably, as evident from similar situations such as the CryptoKitties fiasco [106].

Off-chain solutions are designed to provide instant finality, reduce transaction fees, and improve the transaction throughput; however, we established in [31] that the utilization of payment channels does not provide benefits in the case of local energy trading where the trading relationship between households is ephemeral. One solution would be establishing channels with typical households participating in the P2P trading market. Payment channels require depositing collateral for each recipient a user wants to transact with; additionally, this collateral has to be constantly replenished since it is a means of remuneration. This requirement makes the utilization of payment channels expensive and consequently results in many users' money being locked up. Hence, this is not a viable option. An extension of channels, i.e., payment channel networks, is used to solve the significant collateral lock-up issue of having several payment channels. However, utilizing payment channel networks results in complicated and fee-contingent routing topologies between various peers in the payment network. Moreover, any crash of an intermediate node in the routing topology requires the computation of a new route on the fly.

These fee-contingent routing topologies can be avoided by using another payment channel extension called payment channel hubs. However, operating a hub that assures atomic payments requires high liquidity. For example, imagine if a hub operator maintains ten thousand incoming channels, each with a thousand dollars worth of liquidity, they must deposit a collateral of ten million dollars. Such liquidity requirements make payment hubs highly centralized since only large organizations can afford to lock up ten million dollars. A workaround to address these liquidity issues is commit chains, e.g., Nocust [69], shifting the problem from liquidity to the reliance on *eventual finality*, i.e., the transaction is eventually confirmed to be a part of the canonical state; hence, they do not address the main problem of high transaction confirmation latency.

Fixing the liquidity problem of payment hubs without relying on eventual finality and maintaining privacy guarantees makes the hub-based architecture a viable option for settlements in a P2P trading market. Additionally, such a solution would be suitable for payments in retail markets where the relationships between customers and merchants are ephemeral, and payments are unidirectional, i.e., from a customer to a merchant.

As mentioned before, a distinguishing feature of P2P energy trading markets is their ephemeral nature; more specifically, households are matched based on their demand forecasts, and these forecasts are volatile and result in different pairs of households being matched in various periods.

The problems with existing off-chain solutions, as discussed so far in this section, aggravate in ephemeral markets; e.g., the issue of collateral depletion in payment channel networks [21] (a problem that extends to payment hubs) is aggravated in retail markets since the customers use up all their collateral when paying the merchants due to the unidirectional nature of payments.

Thus, markets with ephemeral relationships between payers and payees are challenging to scale by utilizing state-of-the-art scaling solutions, and a new solution was needed to satisfy our requirements.

Reusable, Instant and Private Payment Guarantees for Cryptocurrencies[12]: If one carefully inspects the arguments made since the beginning of Sect. 3.3.3, the requirements for a low-latency payments solution in markets with ephemeral relationships become evident. These requirements are listed as follows:

1. *Trustless Hub-like Architecture*: An efficient way to handle ephemeral payment relationships is by establishing a Hub routing payments from a sender to any receiver amongst a large set. In the case of retail markets, this payment could go from a customer to any merchant in a consortium of merchants without requiring the customer to establish a payment channel directly with each of them. Additionally, such intermediate hubs should not be able to reveal private information about the participants and not become a single point of failure (read: they should be distributed).
2. *No Liquidity Requirement*: As established before, liquidity requirements imply a strict participation barrier that exhibits a permissioned nature. Thus, the parties collaborating to form the intermediate Hub should not be required to deposit any collateral.
3. *Reusable Collaterals*: The deposited collateral in this solution should act like collateral, not the primary means of remuneration for payees. More specifically, as long as the payer behaves honestly, this collateral should never be depleted.
4. *Decoupling Payment Guarantee from Payment Delivery*: A way to achieve this collateral reusability is by decoupling payment guarantees from payment delivery, i.e., performing high-latency payments on the main blockchain while instantly guaranteeing that the payment will be successful. This decoupling mimics the traditional banking architecture where card transactions are authorized instantly; however, it takes more than a day for the merchant to receive money in their account.

In our paper called "Reusable, Instant, and Private Payment Guarantees for Cryptocurrencies" [82], we proposed a generic architecture that satisfies these

[12]This section reuses text from [82].

requirements in an entirely distributed fashion. While designing our architecture, we faced two challenges. First, our construction must utilize a proactive double-spending protection mechanism that can *selectively* reveal information about dishonest parties while keeping all information about honest parties private. Second, since the payment latency needs to be low, the solution must enable payers to prove vital information privately yet very efficiently.

We address the first challenge by proposing a novel variant of threshold encryption, which we hereon refer to as randomness-reusable threshold encryption (rrTE): it is a threshold encryption scheme with the additional property of revealing the plaintext if two ciphertexts are encrypted using the same randomness. We use rrTE in combination with Pseudo-Random Functions (PRFs) to allow the statekeepers to preclude double-spend attempts without knowing any transaction details, yet still able to reveal vital information in the case of double-spent transactions. The second challenge is addressed by a combination of Threshold Structure-Preserving Signatures (TSPS) [43] and Non-Interactive Zero-Knowledge Proofs (NIZKs). The compatibility of TSPS with efficient NIZKs enables a payer to provide necessary information to payees in our system in zero knowledge without impacting the latency of transactions.

Participants. To explain how combining these cryptographic primitives solves the requirements mentioned above, we introduce the participants of our construction as follows: (1) customers willing to purchase products/services using their cryptocurrencies while expecting low latency; (2) an established consortium of merchants willing to accept fast cryptocurrency payments and (3) statekeepers who are selected from the merchant consortium. Additionally, we include a group of authorities *trusted* for registration of users and verification of collaterals. Although authorities have greater power during the setup phase, they *do not* have any access to future transactions between a customer and a merchant.

Interaction of Participants. In our construction, the participants function as follows: each *customer* in our systems owns collaterals uniquely linked to secret PRF keys. These secret PRF keys are signed by the group of *authorities* utilizing TSPS after they confirm its existence on the smart contract deployed by the system.

The PRF is combined with our rrTE to set up our double-spending protection. By using this signed secret PRF key, each customer generates a random string, which they then use in combination with the public key of the *target merchant* to encrypt their secret identity. For encryption, they use rrTE, which reveals the secret identity of a malicious customer (plaintext) when they attempt to double-spend. More precisely, double-spending in our system amounts to certifying multiple transactions with the same collateral, by which rrTE reveals the dishonest customer's identity. Any *statekeeper* receiving two encrypted ciphertexts using the same random string can catch double-spending and reveal the perpetrator. However, honest customers who only certify one transaction per collateral remain unaffected.

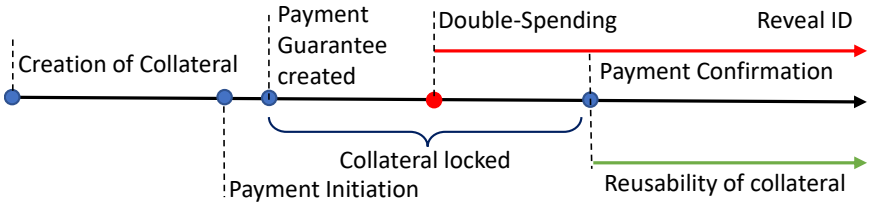


Figure 3.6: Timeline of a transaction in our overlay network

Next, the customer needs to convince the target merchant about the validity of their collateral. However, they must do this without revealing any identifying details. We utilize TSPS and NIZKs to achieve this. The payment guarantee is an indirect proof of collateral by proving knowledge of the TSPS provided to them by the authorities. In the case of a double-spend attempt, this payment guarantee is sufficient for the merchant to be reimbursed through the collateral. This can be done efficiently by carefully selecting the appropriate ZKP systems.

Finally, our architecture also allows a customer to *reuse* their collaterals after a specific time interval as illustrated in Fig. 3.6, collateral is considered *locked* for the period between the customer sending a payment guarantee to the merchant and the actual confirmation of their blockchain transaction. However, the customer's collateral can be reused after confirming a blockchain transaction. Note that this is similar to the execution of a multi-hop payment on payment channel networks that rely on HTLCs, locking up the collateral for a similar time period [99, 20]. Thus, using collateral twice in quick succession is treated as a double-spend and can be detected, but it may be reused once the locking period of the collateral has elapsed.

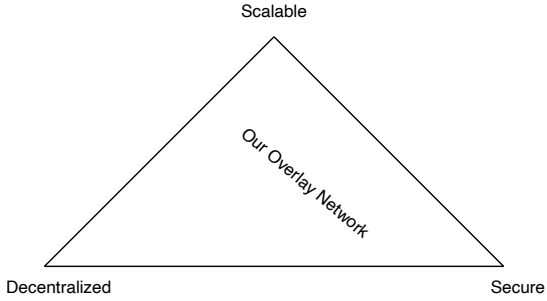


Figure 3.7: Our overlay network in the Scalability Trilemma

As depicted in Fig. 3.7, our overlay network lies in a "sweet spot" in the Scalability Trilemma, i.e., it operates in a hub architecture where we *assume* that the majority

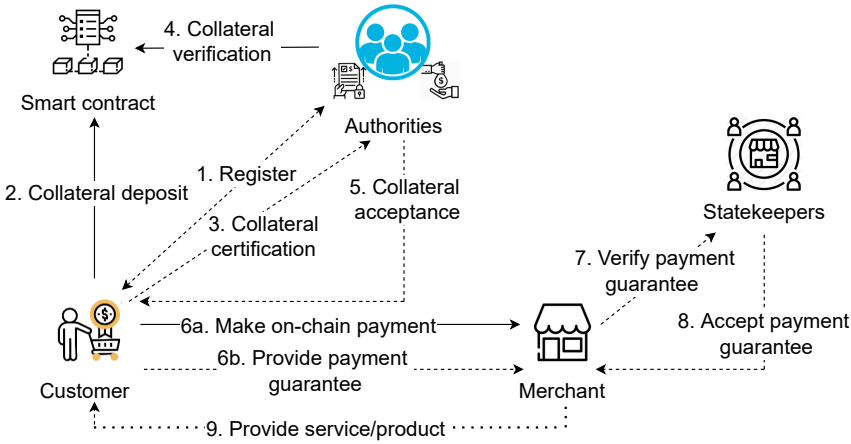


Figure 3.8: Construction of our overlay network

of authorities behave *honestly*. It is well-known that payment channels eventually converge to a centralized hub architecture [126], and while designing our overlay network we intentionally used this to our advantage. We started off with an architecture that utilizes a hub, and removed the need of involvement from the authorities that form this hub during actual payments. Hence, we limited their participation to only the *bootstrapping* and *dispute resolution* phases, conforming to our resolution of maintaining the privacy of honest payers while providing instant payment guarantees. Thus, we position our work more towards *scalability* and *security* in Fig. 3.7.

Construction Overview. Fig. 3.8 illustrates our construction. To explain it, we assume an established set of authorities and a fixed merchant consortium. The payment protocol between a customer and a merchant has off-chain and on-chain components depicted in Fig. 3.8 with dotted and solid arrows, respectively. The protocol proceeds as follows:

1) The customer begins by registering themselves with a set of authorities. **2)** The customer deposits collateral in the smart contract, controlled by this set of authorities. **3)** Once the deposit is confirmed on the blockchain, the customer requests a collateral certification necessary to generate payment guarantees for merchants. **4)** Upon receiving the certification request of the customer, the authorities check the smart contract to confirm if the customer deposited collateral. **5)** The authorities provide the customer with a signed collateral certification. **6a)** During a transaction in the retail market, the customer makes a payment to the target merchant using the underlying cryptocurrency. This is done by sending a payment to the merchant through the smart contract. **6b)** The customer simultaneously generates an encrypted *payment guarantee* by using the collateral certification and sends this to the target merchant along with the transaction identifier of their cryptocurrency payment. **7)** The target merchant forwards

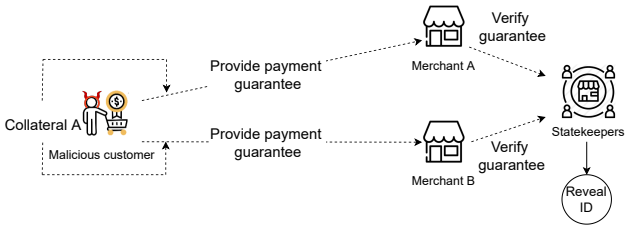


Figure 3.9: Proactive double-spending detection

this encrypted *payment guarantee* to the statekeepers, who individually confirm that it is not a double-spend attempt. The statekeepers compare the received guarantee with each guarantee they receive within a fixed period. If none of these comparisons reveal the plaintext, i.e., the secret identity of the customer, they are guaranteed its uniqueness. Note that this verification can be done using our proposed rrTE and does not require knowing a customer's identity. **8)** Each statekeeper returns a signed payment guarantee to the merchant. **9)** If a majority of the statekeepers return a confirmation, the merchant aggregates these signatures and accepts the payment guarantee. Consequently, the merchant provides the customer with the necessary services/products.

Double-Spending. Double-spending in our construction is a scenario where a malicious customer wants to use their payment guarantee twice within the same locking period, i.e., use the same collateral to promise payments to two distinct merchants. In this case, when the statekeepers receive these guarantees from two different merchants, they can combine them and reveal the secret identity of the cheating customer. A secret to the customer's collateral is revealed along with this identity. The victim merchant uses this secret for remuneration, as the other merchant receives this payment through the underlying blockchain. This is depicted in fig. 3.9.

Background and information on the utilized cryptographic primitives such as CKG, PRF, rrTE, TSPS, and NIZK is given in App. A.

Chapter 4

Conclusion and Future Work

4.1 Conclusion

Blockchain technology has risen to fame as the harbinger of Web3, a version of the Internet that bypasses the efficient but controversial centralized architecture utilized by the present version. Blockchains face a very interesting Trilemma stipulating that a blockchain cannot simultaneously be decentralized, scalable, and secure. This Trilemma is commonly known as the Scalability Trilemma, and new blockchain-based systems that claim to solve it emerge frequently.

This thesis takes a critical stance toward the Scalability Trilemma in blockchains. It dives into this topic with an explorative outlook to understand the trade-offs, investigates existing designs, and eventually proposes improved architectures to help future designers of decentralized applications.

We started by discussing Sc2Share and how this system decentralizes a well-known sharing economy use case, i.e., car sharing, and documents the trade-offs involved. Interestingly, these trade-offs directly correlate to the Scalability Trilemma.

Our study on Storj network investigated the Scalability Trilemma in existing sharing economy use cases, in this case, decentralized storage; we uncovered a security vulnerability in their architecture and reinforced the need to carefully design decentralized systems to find a sweet spot in this Trilemma.

We then focused on the energy domain due to the rich literature suggesting decentralizing its various use cases. Our study on the application of state channels in the energy domain addressed the frequently ignored scalability issues in the literature.

It utilized a novel methodology to classify the use cases based on the applicability of state channels. We concluded that state channels could be applied to optimize wholesale energy trading, EV charging, and electricity consumption optimization. We also observed that the widespread use case of P2P energy trading could not be scaled using existing solutions to address the Trilemma.

Hence, to design a scalable, decentralized, and secure solution for P2P energy trading, we utilized a business model matrix to identify the most critical uncertainties in the P2P energy trading markets and defined four scenarios based on the level of user involvement and customer ownership. We then performed a security and privacy threat analysis on each scenario and specified security and privacy requirements. Using the lessons learned in this study, we proposed fully privacy-preserving, secure, and decentralized billing models for P2P energy trading that, similar to Storj, only require a blockchain for settlements.

Finally, to provide instant settlements, we proposed an overlay network that allows instant confirmations of cryptocurrency payments while ensuring the unlinkability of a users transactions and security against double-spending; in addition, the system provides payment guarantees in less than a second.

4.2 Discussion

Over the last five years, our insights related to the research done during this thesis have evolved. In Sc2Share, we claimed that the decentralized car sharing protocol we developed and implemented is efficient and ready for practical use. However, since then, various events such as the CryptoKitties fiasco [106] and the confirmation latency of Ethereum increasing four-fold [53] have made us realize that this is not the case.

To wit, the deployment cost of Sc2Share on Ethereum was $0.22 when we published our research; now, this cost has gone up to $11.87. The solutions that exist to reduce this cost do not yet solve the Scalability Trilemma; they either sacrifice decentralization or security for scalability.

As we explain in Chapter 3, the design of our fully privacy-preserving billing models and overlay network tries to find a “sweet spot” in the Scalability Trilemma. By utilizing advanced cryptography such as ZKPs and FHE in our protocols, we can utilize a stratified topology while ensuring that the sensitive information of users is never revealed to *any* third party (with an exception of payers who double-spend in our overlay network). Hence, despite relying on certain players in our protocol *more* than on others, we provide security by removing a single point of failure.

Furthermore, our billing models assume that the domain of P2P energy trading is where excess electricity production can be settled. However, the energy domain has

various applications, such as flexibility markets, where this excess can be settled; this alternative is presently being used.

Following the research done in this thesis, the Scalability Trilemma seems to be an omnipresent constraint in systems that utilize blockchain technology. Efforts to improve scalability often reduce security, more robust systems are frequently inefficient and do not scale well, and any effort made to improve either scalability or security usually impacts decentralization.

Moving from a monolithic to a modular architecture often improves scalability and efficiency but usually imposes an entry barrier. For instance, (i) participating in a payment channel network or operating a payment channel hub imposes liquidity requirements, and (ii) rollup operators most often need to have higher/faster computation capabilities than the other participants.

We conjecture that facing the Trilemma from its source, i.e., the L1, is a better approach. Recent blockchains such as Mina [29], and DAGs such as Sui [28] and Aptos [75] are examples of innovative approaches to tackle this Trilemma. Mina utilizes a *succinct* blockchain, i.e., a blockchain whose verification complexity is largely independent of chain length. Mina's blockchain only comprises ZKPs that prove valid state transitions. Anyone accessing Mina's succinct blockchain can verify its correctness, thus providing decentralization and security. Additionally, in Mina, the total work done by validators to produce blocks is split amongst various parties, thus increasing efficiency and, consequently, transaction throughput.

Sui and Aptos utilize a novel DAG-based mempool called Narwhal [45], which separates the task of reliable transaction dissemination from transaction ordering and enables a high-performance consensus. Although guaranteeing the security of DAG networks under such high-throughput settings is still under debate [123], the separation of data dissemination and consensus is a step in the right direction.

4.3 Future Work

The field of blockchain technology and the plethora of decentralized applications it enables will remain a source of innovation. The work done in this thesis was built on foundations laid down by others, and it hopes to serve as a foundation that future researchers can build on. Hence, the innovation efforts do not conclude here, and we envision the following future studies.

A fully privacy-preserving and scalable car sharing solution: blockchain technology has enabled substantial progress in modern cryptography, with various methods for computing over private data such as MPC, ZKPs and FHE becoming more and more efficient due to the unique use cases and strong focus offered by

decentralized applications and cryptocurrencies. These advancements, in addition to a modular architecture, have given rise to modified EVMs such as zkEVMs [98] and fhEVMs [127] substantially reducing transaction fees for applications deployed on Ethereum, increasing throughput, and in the case of fhEVM, providing fully privacy-preserving applications.

Hence, we aim to use these advances in cryptography to propose a fully privacy-preserving and scalable car-sharing solution deployed on a permissionless blockchain. Sc2share presents only a basic architecture that can provide a foundation for modular improvements to break the link between a renter and rentee completely, in addition to preserving the anonymity of users. In addition, similar to our billing models and the architecture of solutions such as Storj, this modular approach can also perform all necessary computation off-chain while only utilizing the blockchain for settlements of rental contracts. This allows offloading the transaction load from the main blockchain and improving scalability.

Instant guarantees for generic computations: our idea of decoupling guarantees from delivery only helps improve the latency of monetary transactions, e.g., in retail markets where a customer does not want to wait an hour for their ice cream. However, we believe this decoupling can also be extended to generic computations. This is an exciting future work venue because state-of-the-art general-purpose scaling solutions often ignore transaction confirmation latency due to their focus on increasing the throughput and decreasing the transaction fees. These priorities help improve the usability of blockchain technology; however, we believe that the solutions to enhance throughput have optimized it enough and that the real performance bottleneck nowadays is latency. Thus, more research is needed to provide a faster settlement finality for these transactions. Hence, this work will help address the primarily ignored problem of finality in state-of-the-art general-purpose scaling solutions.

Rollup centralization: introducing solutions based on a rollup architecture has revolutionized scalability in blockchains. Rollups allow for incredible throughput gains without requiring design changes for the underlying blockchain. In the case of ZK-Rollups, these advantages come at the cost of increased computation for the rollup operators due to the requirement of computing a ZKP for proving a valid state transition that is a byproduct of all the transactions present in a rollup batch.

This increased computation imposes a higher restriction on becoming a rollup operator, thus becoming a bottleneck in the decentralization of a blockchain. Additionally, the recent efforts to develop hardware that accelerates proof generation for ZKP may lead to additional centralization and security concerns (similar to the concerns of security in blockchains that use Intel SGX [117]).

Thus, solutions for the scalability of blockchains that does not raise concerns of a negative impact on decentralization and security is an exciting venue for future

research.

Fast confirmation for all transactions: recently, blockchains such as Sui [28] incorporated a "fast path", i.e., a way for *non-conflicting* transactions to achieve faster confirmation as compared to the transactions that require consensus for being sequenced. Token transfers, NFT-specific transactions can all be classified as non-conflicting transactions that achieve nearly instant finality (300 ms in Sui); however, the transactions that require consensus still have a longer finality time.

In Sui, the execution model for enabling fast-path transactions actually *adds* to the latency of transactions that require consensus due to a particular "locking" mechanism to address the double-spending issue in fast-path transactions. This is somewhat controversial, as it effectively slows down all complex transactions in the system.

Thus, a solution to quickly confirm all transactions would be desirable.

Incentive mechanisms in DAG-based systems: the upsurge in DAG-based blockchain systems such as Prism [22], OHIE [125], Kaspa [107], and Sui [28] has come with claims that the Scalability Trilemma has been completely addressed [4]. This claim has not proven sturdy enough, with a recent paper experimentally proving reduced security in a high-throughput setting in the case of Prism and OHIE [123].

Additionally, there are no precise incentive mechanisms for the operators of these DAG-based systems. The total transaction fees accumulated in each round are supposedly split between all active operators for that round, leading to doubts whether this incentive is sufficient to ensure honest behavior.

Thus, a study on incentive schemes utilized in these DAG-based systems is crucial and can be a solid path for future work.

Bibliography

[1] Firewall Times. Uber data breaches: Full timeline through 2022. https://firewalltimes.com/uber-data-breach-timeline/. Accessed: 2023-01-13.

[2] FirstPost. Explained: what caused the Instagram outage and why are Meta's services going down so often. https://www.firstpost.com/tech/news-analysis/what-caused-the-instagram-outage-and-why-are-metas-services-going-down-so-often-11549061.html/. Accessed: 2023-01-12.

[3] Grand View Research. B2C e-commerce market worth $7.6 trillion by 2028. https://www.grandviewresearch.com/press-release/global-b2c-e-commerce-market. Accessed: 2022-12-27.

[4] Kaspa Network. https://kaspa.network/. Accessed: February 14, 2024.

[5] RIPEMD. https://en.wikipedia.org/wiki/RIPEMD. Accessed: March 11, 2024.

[6] SHA-2. https://en.wikipedia.org/wiki/SHA-2. Accessed: March 11, 2024.

[7] Statista. Etsy active buyers. https://www.statista.com/statistics/409375/etsy-active-buyers/. Accessed: 2022-11-11.

[8] Statista. Europe: visits to top e-commerce sites 2022. https://www.statista.com/statistics/1338633/retail-websites-europe/. Accessed: 2022-11-22.

[9] Statista. Facebook monthly active users. https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/. Accessed: 2022-11-11.

[10] Statista. Global Facebook advertising revenue forecast 2026. https://www.statista.com/statistics/544001/facebooks-advertising-revenue-worldwide-usa/. Accessed: 2022-12-27.

[11] Statista. Google: advertising revenue 2021. https://www.statista.com/statistics/266249/advertising-revenue-of-google/. Accessed: 2022-12-27.

[12] Statista. Most visited online retail websites worldwide in 2022, by monthly traffic. https://www.statista.com/statistics/274708/online-retail-and-auction-ranked-by-worldwide-audiences/. Accessed: 2022-11-22.

[13] Statista. Number of compromised data records in selected data breaches as of August 2022 (in millions). https://www.statista.com/statistics/290525/cyber-crime-biggest-online-data-breaches-worldwide/. Accessed: 2022-12-27.

[14] Aysajan Abidin, Abdelrahaman Aly, Sara Cleemput, and Mustafa A Mustafa. An MPC-based privacy-preserving protocol for a local electricity trading market. In *Int. Conf. on Cryptology and Network Security*, pages 615–625, 2016.

[15] Lawrence Abrams. Amazon AWS outage shows data in the cloud is not always safe. https://www.bleepingcomputer.com/news/technology/amazon-aws-outage-shows-data-in-the-cloud-is-not-always-safe/, 2019.

[16] Airbnb. Airbnb service fees - Airbnb help center. https://www.airbnb.com/help/article/1857. Accessed: 2022-11-23.

[17] Ross J. Anderson. The Eternity Service. https://www.cl.cam.ac.uk/~rja14/eternity/eternity.html/, 1996.

[18] Merlinda Andoni, Valentin Robu, David Flynn, Simone Abram, Dale Geach, David Jenkins, Peter McCallum, and Andrew Peacock. Blockchain technology in the energy sector: A systematic review of challenges and opportunities. *Renewable and Sustainable Energy Reviews*, 100:143–174, 2019.

[19] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed Cocco, and Jason Yellick. Hyperledger Fabric: A distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, EuroSys '18, New York, NY, USA, 2018. Association for Computing Machinery.

[20] Lukas Aumayr, Kasra Abbaszadeh, and Matteo Maffei. Thora: Atomic and privacy-preserving multi-channel updates. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 165–178, New York, NY, USA, 2022. Association for Computing Machinery.

[21] Zeta Avarikioti, Krzysztof Pietrzak, Iosif Salem, Stefan Schmid, Samarth Tiwari, and Michelle Yeo. Hide & Seek: Privacy-preserving rebalancing on payment channel networks. In *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers*, page 358–373, Berlin, Heidelberg, 2022. Springer-Verlag.

[22] Vivek Bagaria, Sreeram Kannan, David Tse, Giulia Fanti, and Pramod Viswanath. Prism: Deconstructing the blockchain to approach physical limits. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 585–602, New York, NY, USA, 2019. Association for Computing Machinery.

[23] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. Consensus in the Age of Blockchains. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 183–198, New York, NY, USA, 2019. ACM.

[24] Juan Benet. IPFS - Content Addressed, Versioned, P2P File System (DRAFT 3). Technical report, IPFS, 2014.

[25] Iddo Bentov, Pavel Hubáček, Tal Moran, and Asaf Nadler. Tortoise and Hares Consensus: The Meshcash framework for incentive-compatible, scalable cryptocurrencies. In Shlomi Dolev, Oded Margalit, Benny Pinkas, and Alexander Schwarzmann, editors, *Cyber Security Cryptography and Machine Learning*, pages 114–127, Cham, 2021. Springer International Publishing.

[26] Iddo Bentov, Julian Loss, Tal Moran, and Barak Shani. The SpaceMesh protocol: Tortoise and Hare protocols in Space. https://spacemesh.io/blog/spacemesh-white-paper-1/, September 2019.

[27] Immutable Record: The Blockchain Before Bitcoin. Haber & Stornetta. https://immutablerecord.com/the-co-inventors/. Accessed: 2023-04-04.

[28] Sam Blackshear, Andrey Chursin, George Danezis, Anastasios Kichidis, Lefteris Kokoris-Kogias, Xun Li, Mark Logan, Ashok Menon, Todd Nowacki, Alberto Sonnino, Brandon Williams, and Lu Zhang. Sui Lutris: A blockchain combining broadcast and consensus. https://doi.org/10.48550/arXiv.2310.18042/, 2023. arXiv:2310.18042.

[29] Joseph Bonneau, Izaak Meckler, Vanishree Rao, and Evan Shapiro. Coda: Decentralized cryptocurrency at scale. https://eprint.iacr.org/2020/352, 2020. Cryptology ePrint Archive, Paper 2020/352.

[30] Richard Gendal Brown, James Carlyle, Ian Grigg, and Mike Hearn. Corda: An introduction. https://docs.r3.com/en/pdf/corda-introductory-whitepaper.pdf/, August 2016.

[31] Clemens Brunner, Akash Madhusudan, Dominik Engel, and Bart Preneel. Off-chain state channels in the energy domain. In *2021 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pages 1–5, 2021.

[32] V. Buterin. A next-generation smart contract and decentralized application platform. https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf/. last checked on 2023-09-27.

[33] Vitalik Buterin. An incomplete guide to rollups. https://vitalik.ca/general/2021/01/05/rollup.html, 2021.

[34] Vitalik Buterin. Paths toward single-slot finality. https://notes.ethereum.org/@vbuterin/single_slot_finality/, November 2022.

[35] Carole Cadwalladr and Emma Graham-Harrison. Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach. https://www.theguardian.com/news/2018/mar/17/cambridge-analytica-facebook-influence-us-election, 2018.

[36] J. Alejandro F. Castellanos, Debora Coll-Mayor, and José Antonio Notholt. Cryptocurrency as Guarantees of Origin: Simulating a Green Certificate Market with the Ethereum Blockchain. In *5th IEEE International Conference on Smart Energy Grid Engineering*, pages 367–372, Oshawa, Canada, 2017. IEEE.

[37] Michael Castillo Del. Nasdaq explores how blockchain could fuel solar energy market. https://www.coindesk.com/markets/2016/05/18/nasdaq-explores-how-blockchain-could-fuel-solar-energy-market/, 2016. Accessed: 2023-08-08.

[38] Miguel Castro and Barbara Liskov. Practical Byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, page 173–186, USA, 1999. USENIX Association.

[39] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437, Cham, 2017. Springer International Publishing.

[40] Bram Cohen. Bittorrent protocol specification. https://www.bittorrent.org/beps/bep_0003.html. Accessed: 2023-03-10.

[41] Keith Collins. A running list of websites and apps that have banned, blocked, deleted, and otherwise dropped white supremacists. https://qz.com/1055141/what-websites-and-apps-have-banned-neo-nazis-and-white-supremacists/, last checked on 2023-05-08, 2018.

[42] Consensys. Grid+: Welcome to the future of energy. https://gridplus.io/whitepaper, 2017. accessed on April 2020.

[43] Elizabeth Crites, Markulf Kohlweiss, Bart Preneel, Mahdi Sedaghat, and Daniel Slamanig. Threshold structure-preserving signatures. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023*, pages 348–382, Singapore, 2023. Springer Nature Singapore.

[44] Mariana Botelho da Gama, John Cartlidge, Antigoni Polychroniadou, Nigel P. Smart, and Younes Talibi Alaoui. Kicking-the-bucket: Fast privacy-preserving trading using buckets. In *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers*, page 20–37, Berlin, Heidelberg, 2022. Springer-Verlag.

[45] George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. Narwhal and Tusk: a DAG-based mempool and efficient BFT consensus. In *Proceedings of the Seventeenth European Conference on Computer Systems*, EuroSys '22, page 34–50, New York, NY, USA, 2022. Association for Computing Machinery.

[46] Sammy de Figueiredo, Akash Madhusudan, Vincent Reniers, Svetla Nikova, and Bart Preneel. Exploring the Storj network: a security analysis. In Chih-Cheng Hung, Jiman Hong, Alessio Bechini, and Eunjee Song, editors, *SAC '21: The 36th ACM/SIGAPP Symposium on Applied Computing, Virtual Event, Republic of Korea, March 22-26, 2021*, pages 257–264. ACM, 2021.

[47] Christian Decker and Roger Wattenhofer. A fast and scalable payment network with Bitcoin duplex micropayment channels. In *Symposium on Self-Stabilizing Systems*, pages 3–18. Springer, 2015.

[48] Dgtlinfra. Facebook's 18 data centers: $20bn investment, 40m square feet. https://dgtlinfra.com/facebook-18-data-centers-20bn-investment/. Accessed: 2022-12-27.

[49] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, page 21, USA, 2004. USENIX Association.

[50] Viktorija Dudjak, Diana Neves, Tarek Alskaif, Shafi Khadem, Alejandro Pena-Bello, Pietro Saggese, Benjamin Bowler, Merlinda Andoni, Marina Bertolini,

Yue Zhou, et al. Impact of local energy markets integration in power systems layer: A comprehensive review. *Applied Energy*, 301:117434, 2021.

[51] Elizabeth C Economy. The great firewall of China: Xi Jinping's internet shutdown. https://www.theguardian.com/news/2018/jun/29/the-great-firewall-of-china-xi-jinpings-internet-shutdown/, 2018.

[52] Ethereum. Remix IDE. https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.18+commit.87f61d96.js, last checked on 2023-08-21, 2016.

[53] ethos.dev. The beacon chain Ethereum 2.0 explainer you need to read first. https://ethos.dev/beacon-chain. Accessed: 2023-05-10.

[54] Filecoin. Filecoin is a decentralized storage network designed to store humanity's most important information. https://filecoin.io/, 2020.

[55] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin Backbone protocol: Analysis and applications. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 281–310, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[56] Joseph Thachil George. *Introducing Blockchain Applications: Understand and Develop Blockchain Applications Through Distributed Systems*. Apress, Berkeley, CA, 2022. ISBN: 978-1-4842-7480-4, DOI: 10.1007/978-1-4842-7480-4_3.

[57] Evangelos Georgiadis. How many transactions per second can Bitcoin really handle? Theoretically. https://eprint.iacr.org/2019/416/, 2019. Cryptology ePrint Archive, Report 2019/416.

[58] Naveen Goud. Top 5 cloud security related data breaches! https://www.cybersecurity-insiders.com/top-5-cloud-security-related-data-breaches/, 2017.

[59] Ian Grigg. EOS: An introduction. https://www.iang.org/papers/EOS_An_Introduction-BLACK-EDITION.pdf/, 2017.

[60] Stuart Haber and W. Scott Stornetta. How to time-stamp a digital document. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology-CRYPTO' 90*, pages 437–455, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[61] Wendy Hall and Thanassis Tiropanis. Web evolution and web science. *Computer Networks*, 56(18):3859–3865, 2012. The WEB we live in.

[62] Kazi Nazmul Hasan, Robin Preece, and Jovica V Milanović. Existing approaches and trends in uncertainty modelling and probabilistic stability analysis of power systems with renewable generation. *Renewable and Sustainable Energy Reviews*, 101:168–180, 2019.

[63] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. Tumblebit: An untrusted Bitcoin-compatible anonymous payment hub. In *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017*. The Internet Society, 2017.

[64] Robert A Heinlein. *The moon is a harsh mistress*. New York : Putnam, 1966. ISBN: 9780340837948, ISBN10: 0340837942, Pages: 288.

[65] Kashmir Hill. How Target figured out a teen girl was pregnant before her father did. https://www.forbes.com/sites/kashmirhill/2012/02/16/how-target-figured-out-a-teen-girl-was-pregnant-before-her-father-did/?sh=53d927356668. Accessed: 2022-08-30.

[66] ISO. ISO/IEC 18014. https://en.wikipedia.org/wiki/ISO/IEC_18014/, 2008. accessed on Feb 2024.

[67] Simon Josefsson and Ilari Liusvaara. Edwards-Curve Digital Signature Algorithm (EdDSA). RFC 8032 https://www.rfc-editor.org/info/rfc8032, January 2017.

[68] Ghassan O. Karame, Elli Androulaki, and Srdjan Capkun. Double-spending fast payments in Bitcoin. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, page 906–917, New York, NY, USA, 2012. Association for Computing Machinery.

[69] Rami Khalil, Alexei Zamyatin, Guillaume Felley, Arthur Gervais, and Pedro Moreno-sanchez. Commit-Chains: Secure, Scalable Off-Chain Payments. https://eprint.iacr.org/2018/642.pdf/, 2018.

[70] Fabian Knirsch, Clemens Brunner, Andreas Unterweger, and Dominik Engel. Decentralized and Permission-less Green Energy Certificates with GECKO. *Energy Informatics*, 3(2):1–17, 2020.

[71] Fabian Knirsch, Oliver Langthaler, and Dominik Engel. Trust-less Electricity Consumption Optimization in Local Energy Communities. *Energy Informatics*, 2(1):1–12, 2019.

[72] Fabian Knirsch, Andreas Unterweger, Günther Eibl, and Dominik Engel. Privacy-Preserving Smart Grid Tariff Decisions with Blockchain-Based Smart Contracts. In Wilson Rivera, editor, *Sustainable Cloud and Energy Services:*

Principles and Practices, chapter 4, pages 85–116. Springer International Publishing, Cham, Switzerland, 2017.

[73] Fabian Knirsch, Andreas Unterweger, and Dominik Engel. Privacy-preserving Blockchain-based Electric Vehicle Charging with Dynamic Tariff Decisions. *Journal on Computer Science - Research and Development (CSRD)*, 33(1):71–79, 2018.

[74] Weicong Kong, Zhao Yang Dong, David J Hill, Fengji Luo, and Yan Xu. Short-term residential load forecasting based on resident behaviour learning. *IEEE Transactions on power systems*, 33(1):1087–1088, 2017.

[75] Aptos Labs. The Aptos blockchain: Safe, scalable, and upgradeable Web3 infrastructure. https://aptos.dev/assets/files/Aptos-Whitepaper-47099b4b907b432f81fc0effd34f3b6a.pdf, 2022.

[76] Storj Labs. Storj: A decentralized cloud storage network framework (v3.0). https://storj.io/storj.pdf/, 2018.

[77] Leslie Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169, May 1998.

[78] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.

[79] Yehuda Lindell. Secure multiparty computation. *Commun. ACM*, 64(1):86–96, Dec 2020.

[80] Xin Liu, Xiaowei Yang, and Yong Xia. NetFence: Preventing Internet Denial of Service from inside out. *SIGCOMM Comput. Commun. Rev.*, 40(4):255–266, August 2010.

[81] Akash Madhusudan, Mahdi Sedaghat, Philipp Jovanovic, and Bart Preneel. Nirvana: Instant and anonymous payment-guarantees. Cryptology ePrint Archive, Paper 2022/872, 2022. https://eprint.iacr.org/2022/872.

[82] Akash Madhusudan, Mahdi Sedaghat, Samarth Tiwari, Kelong Cong, and Bart Preneel. Reusable, instant and private payment guarantees for cryptocurrencies. In Leonie Simpson and Mir Ali Rezazadeh Baee, editors, *Information Security and Privacy*, pages 580–605, Cham, 2023. Springer Nature Switzerland.

[83] Akash Madhusudan, Iraklis Symeonidis, Mustafa A. Mustafa, Ren Zhang, and Bart Preneel. Sc2share: Smart contract for secure car sharing. In Paolo Mori, Steven Furnell, and Olivier Camp, editors, *Proceedings of the 5th International Conference on Information Systems Security and Privacy, ICISSP 2019, Prague, Czech Republic, February 23-25, 2019*, pages 163–171. SciTePress, 2019.

[84] Akash Madhusudan, Fairouz Zobiri, and Mustafa A. Mustafa. Billing models for peer-to-peer electricity trading markets with imperfect bid-offer fulfillment. In *2022 IEEE International Smart Cities Conference (ISC2)*, pages 1–7, 2022.

[85] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001.

[86] Esther Mengelkamp, Johannes Gärttner, Kerstin Rock, Scott Kessler, Lawrence Orsini, and Christof Weinhardt. Designing microgrid energy markets: A case study: The Brooklyn Microgrid. *Applied Energy*, 210:870–880, 2018.

[87] Esther Mengelkamp, Benedikt Notheisen, Carolin Beer, David Dauer, and Christof Weinhardt. A blockchain-based smart grid: towards sustainable local energy markets. *Computer Science - Research and Development*, 33(1):207–214, 2018.

[88] Michael Merz. European energy trading firms test peer-to-peer trading over the blockchain. https://enerchain.ponton.de/index.php/21-enerchain-p2p-trading-project/, 2017. accessed on April 2020.

[89] Mehdi Montakhabi, Akash Madhusudan, Shenja van der Graaf, Abuding Aishajiang, Pieter Ballon, and Mustafa Asan Mustafa. Sharing economy in future electricity markets: Security and privacy analysis. In *Workshop on Decentralized IoT Systems and Security (DISS) 2020, co-located with NDSS 2020*. Internet Society, 2020-02-23.

[90] Eric Munsing, Jonathan Mather, and Scott Moura. Blockchains for decentralized optimization of energy resources in microgrid networks. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pages 2164–2171, Mauna Lani, HI, USA, 2017. IEEE.

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

[92] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, USA, 2016.

[93] Nervos. Blockchain Trilemma: The Ultimate Guide. https://www.nervos.org/knowledge-base/blockchain_trilemma. Last Accessed: March 2024.

[94] 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*, ConsensusDay '22, page 43–52, New York, NY, USA, 2022. Association for Computing Machinery.

[95] Eric Newcomer. Uber starts charging what it thinks you're willing to pay. URL: https://www.bloomberg.com/news/articles/2017-05-19/uber-s-future-may-rely-on-predicting-how-much-you-re-willing-to-pay, last checked on 2023-05-08, 2017.

[96] Lily Hay Newman. 1.2 billion records found exposed online in a single server. https://www.wired.com/story/billion-records-exposed-online/, 2019.

[97] James S. Nightingale, Yingjie Wang, Fairouz Zobiri, and Mustafa A. Mustafa. Effect of clustering in federated learning on non-IID electricity consumption prediction. In *2022 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–5, 2022.

[98] Polygon. Polygon zkEVM: Bring Ethereum to everyone. https://polygon.technology/polygon-zkevm, 2023. Accessed: 2023-10-05.

[99] Joseph Poon and Thaddeus Dryja. The Bitcoin lightning network: Scalable off-chain instant payments. https://lightning.network/lightning-network-paper.pdf/, January 2016.

[100] Protos. Bitcoin thieves got away with ATM double-spending spree across Canada. URL: https://protos.com/bitcoin-double-spending-canada-atm-honeybadger-scam-calgary/, last checked on 2023-05-10, 2021.

[101] Deepak Puthal, Nisha Malik, Saraju Mohanty, Elias Kougianos, and Gautam Das. Everything you wanted to know about the blockchain: Its promise, components, processes, and problems. *IEEE Consumer Electronics Magazine*, 7:6–14, 07 2018.

[102] Mayank Raikwar, Danilo Gligoroski, and Katina Kralevska. SoK of used cryptography in blockchain. *IEEE Access*, 7:148550–148575, 2019.

[103] Andrei Vlad Sambra et al. Solid : A platform for decentralized social applications based on linked data. http://emansour.com/research/lusail/solid_protocols.pdf/, 2016.

[104] Caspar Schwarz-Schilling, Joachim Neu, Barnabé Monnot, Aditya Asgaonkar, Ertem Nusret Tas, and David Tse. Three attacks on Proof-of-Stake Ethereum. In Ittay Eyal and Juan Garay, editors, *Financial Cryptography and Data Security*, pages 560–576, Cham, 2022. Springer International Publishing.

[105] Pierluigi Siano, Giuseppe De Marco, Alejandro Rolan, and Vincenzo Loia. A Survey and Evaluation of the Potentials of Distributed Ledger Technology for Peer-to-Peer Transactive Energy Exchanges in Local Energy Markets. *IEEE Systems Journal*, 13(3):3454–3466, 2019.

[106] S. Matthew Smith. The spectacular collapse of Cryptokitties, the first big blockchain game. https://spectrum.ieee.org/cryptokitties/, 2022. accessed on September 2023.

[107] Yonatan Sompolinsky, Shai Wyborski, and Aviv Zohar. PHANTOM GHOSTDAG: a scalable generalization of Nakamoto consensus: September 2, 2021. In *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, AFT '21, page 57–70, New York, NY, USA, 2021. Association for Computing Machinery.

[108] Virendra Soni. Top 6 secure cloud storage providers comparison in 2020. https://wire19.com/best-cloud-storage-providers-comparison/, 2020.

[109] Sia stats. Storage marketplace. https://siastats.info/storage_pricing/, 2020.

[110] Zhou Su, Yuntao Wang, Qichao Xu, Minrui Fei, Yu Chu Tian, and Ning Zhang. A Secure Charging Scheme for Electric Vehicles with Smart Communities in Energy Blockchain. *IEEE Internet of Things Journal*, 2018.

[111] Swarm. Storage and communication for a sovereign digital society. https://ethersphere.github.io/swarm-home/, 2020.

[112] Iraklis Symeonidis, Abdelrahaman Aly, Mustafa A. Mustafa, Bart Mennink, Siemen Dhooghe, and Bart Preneel. SePCAR: A secure and privacy-enhancing protocol for car access provision. *in the 22nd European Symposium on Research in Computer Security (ESORICS'17), ser. LNCS, vol. 10493. Springer, 2017*, pages 475–493, 2017.

[113] Tardigrade.io. Decentralized object storage for developers. https://tardigrade.io/, 2020.

[114] Manny Trillo. Stress test prepares Visanet for the most wonderful time of the year. http://www.visa.com/blogarchives/us/2013/10/10/stress-testprepares-visanet-for-the-most-wonderfultime-of-the-year/index.html, 2013. Accessed: 2023-05-01.

[115] Carmela Troncoso, Marios Isaakidis, George Danezis, and Harry Halpin. Systematizing decentralization and privacy: Lessons from 15 years of research and deployments. *Proceedings on Privacy Enhancing Technologies*, 2017(4):404–426, 2017.

[116] Liam Tung. Google details 'catastrophic' cloud outage events: Promises to do better next time. https://www.zdnet.com/article/google-details-catastrophic-cloud-outage-events-promises-to-do-better-next-time/, 2019.

[117] Stephan van Schaik, Alex Seto, Thomas Yurek, Adam Batori, Bader AlBassam, Christina Garman, Daniel Genkin, Andrew Miller, Eyal Ronen, and Yuval Yarom. SoK: SGX.Fail: How stuff gets eXposed. https://sgx.fail, 2022.

[118] David Vorick and Luke Champine. Sia: Simple decentralized storage. https://sia.tech/sia.pdf/, 2014.

[119] Yingjie Wang, Fairouz Zobiri, Mustafa A Mustafa, James Nightingale, and Geert Deconinck. Consumption prediction with privacy concern: Application and evaluation of federated learning. *Sustainable Energy, Grids and Networks*, 38:101248, 2024.

[120] Wikipedia. Napster. https://en.wikipedia.org/wiki/Napster/, 2001.

[121] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. https://ethereum.github.io/yellowpaper/paper.pdf/, October 2014.

[122] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger Byzantium version. https://ethereum.github.io/yellowpaper/paper.pdf, last checked on 2018-08-5, 2014.

[123] Shichen Wu, Puwen Wei, Ren Zhang, and Bowen Jiang. Security-performance tradeoff in DAG-based Proof-of-Work blockchain protocols. https://eprint.iacr.org/2023/1089/, 2023. Cryptology ePrint Archive, Paper 2023/1089.

[124] Karl Wüst and Arthur Gervais. Do you need a blockchain? https://eprint.iacr.org/2017/375/, 2017. Cryptology ePrint Archive, Paper 2017/375.

[125] Haifeng Yu, Ivica Nikolic, Ruomu Hou, and Prateek Saxena. OHIE: blockchain scaling made simple. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 90–105. IEEE, 2020.

[126] Philipp Zabka, Klaus-T. Foerster, Christian Decker, and Stefan Schmid. Short paper: A centrality analysis of the lightning network. In *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers*, page 374–385, Berlin, Heidelberg, 2022. Springer-Verlag.

[127] Zama. Zama fhEVM: Programmable privacy for the EVM. https://www.zama.ai/fhevm, 2023. Accessed: 2023-10-05.

[128] Ren Zhang and Bart Preneel. On the Necessity of a Prescribed Block Validity Consensus: Analyzing Bitcoin Unlimited Mining Protocol. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '17, page 108–119, New York, NY, USA, 2017. Association for Computing Machinery.

[129] Ren Zhang, Dingwei Zhang, Quake Wang, Shichen Wu, Jan Xie, and Bart Preneel. NC-Max: Breaking the security-performance tradeoff in Nakamoto consensus. In *29th Annual Network and Distributed System Security Symposium, NDSS 2022, San Diego, California, USA, April 24-28, 2022*. The Internet Society, 2022.

[130] Shoshana Zuboff. *The Age of Surveillance Capitalism*. Publicaffairs, 2019. ISBN-13: 978-1-61039-569-4.

Part II

Publications

Publications

International Conferences with Proceedings

1. Akash Madhusudan, Iraklis Symeonidis, Mustafa A. Mustafa, Ren Zhang, and Bart Preneel. Sc2Share: Smart Contract for Secure Car Sharing. In Paolo Mori, Steven Furnell, and Olivier Camp, editors, Proceedings of the 5th International Conference on Information Systems Security and Privacy, ICISSP 2019, Prague, Czech Republic, February 23-25, 2019, pages 163–171. SciTePress, 2019 —-See p. 75.

2. Akash Madhusudan, Fairouz Zobiri, and Mustafa A. Mustafa. Billing Models for Peer-to-Peer Electricity Trading Markets with Imperfect Bid-Offer Fulfillment. In 2022 IEEE International Smart Cities Conference (ISC2), Pafos, Cyprus, 2022, pp. 1-7, doi: 10.1109/ISC255366.2022.9922530. —-See p. 133.

3. Akash Madhusudan, Mahdi Sedaghat, Samarth Tiwari, Kelong Cong, and Bart Preneel. Reusable, Instant and Private Payment Guarantees for Cryptocurrencies. In Information Security and Privacy - 2023 Australasian Conference, ACISP 2023, Brisbane, Australia, Lecture Notes in Computer Science, Springer-Verlag, pp. 580-605, 2023. —-See p. 177.

4. Sammy de Figueiredo, Akash Madhusudan, Vincent Reniers, Svetla Nikova, and Bart Preneel. Exploring the Storj Network: a Security Analysis. In Chih-Cheng Hung, Jiman Hong, Alessio Bechini, and Eunjee Song, editors, SAC '21: The 36th ACM/SIGAPP Symposium on Applied Computing, Virtual Event, Republic of Korea, March 22-26, 202, pages 257–264. ACM, 2021 —-See p. 95.

5. Clemens Brunner, Akash Madhusudan, Dominik Engel, and Bart Preneel. Off-chain State Channels in the Energy Domain. In 2021 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), Washington, DC, USA, 2021, pp. 1-5, doi: 10.1109/ISGT49243.2021.9372246. —-See p. 117.

6. Mehdi Montakhabi, Akash Madhusudan, Shenja van der Graaf, Aysajan Abidin, Pieter Ballon, and Mustafa A. Mustafa. Sharing Economy in Future Electricity Markets: Security and Privacy Analysis. In Workshop on Decentralized IoT Systems and Security (DISS) 2020, co-located with NDSS 2020, San Diego, California, USA. Internet Society, 2020-02-23.

7. Mehdi Montakhabi, Akash Madhusudan, Mustafa A. Mustafa, Wim Vanhaverbeke, Esteve Almirall, and Shenja Van der Graaf. Leveraging Blockchain for Energy Transition in Urban Contexts. Journal of Big Data & Society (BD&S) -(-), 13 pages, 2023.

8. Vincent Reniers, Yuan Gao, Ren Zhang, Paolo Viviani, Akash Madhusudan, Bert Lagaisse, Svetla Nikova, Dimitri Van Landuyt, Riccardo Lombardi, Bart Preneel, and Wouter Joosen. Authenticated and Auditable Data Sharing via Smart Contract. In Proceedings of the 35th Annual ACM Symposium on Applied Computing, SAC '20, page 324–331, New York, NY, USA, 2020. Association for Computing Machinery

9. Mehdi Montakhabi, Shenja van der Graaf, Akash Madhusudan, Roozbeh Sarenche, and Mustafa A. Mustafa. Fostering Energy Transition in Smart Cities: DLTs for Peer-to-Peer Electricity Trading. In 2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS), pages 466–472, 2021

Unpublished Manuscript

10. Akash Madhusudan, Mahdi Sedaghat, Philipp Jovanovic, and Bart Preneel. Nirvana: Instant and Anonymous Payment-Guarantees. Cryptology ePrint Archive, Paper 2022/872, 2022. https://eprint.iacr.org/2022/872

Publication 1

SC2Share: Smart Contract for Secure Car Sharing

Publication Data

Akash Madhusudan, Iraklis Symeonidis, Mustafa A. Mustafa, Ren Zhang, and Bart Preneel. “Sc2share: Smart contract for secure car sharing”. In Paolo Mori, Steven Furnell, and Olivier Camp, editors, Proceedings of the 5th International Conference on Information Systems Security and Privacy, ICISSP 2019, Prague, Czech Republic, February 23-25, 2019, pages 163–171. SciTePress, 2019

Contributions

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

SC2Share: Smart Contract for Secure Car Sharing

Akash Madhusudan, Iraklis Symeonidis, Mustafa A. Mustafa, Ren Zhang, and Bart Preneel

KU Leuven, ESAT/COSIC and imec, Leuven, Belgium and Department of Computer Science, University of Manchester and RISE, Stockholm and Nervos, China
{akash.madhusudan, bart.preneel}@esat.kuleuven.be
{mustafa.mustafa}@manchester.ac.uk
{ren}@nervos.org
{iraklis.symeonidis}@ri.se

Abstract. This paper presents an efficient solution for the booking and payments functionality of a car sharing system that allows individuals to share their personal, underused cars in a completely decentralized manner, annulling the need of an intermediary. Our solution, named SC^{2}Share, leverages smart contracts and uses them to carry out secure and private car booking and payments. Our experiments on SC^{2}Share on the Ethereum testnet guarantee high security and privacy to its users and confirm that our system is cost-efficient and ready for practical use.

1 Introduction

Recent years have witnessed an exponential growth of the sharing economy [5]. It enables individuals to share their personal assets for financial gain, helping others who temporarily need these assets. As of 2018, all practical solutions supporting the sharing economy rely on centralized platforms for exchanging information and settling payments. Unfortunately, a series of incidents indicates that these platforms are not as trustworthy as they promise to be. First, their centralized membership system and pricing scheme allow them to undermine *fairness*. Airbnb, an online marketplace for sharing accommodation, has been in the news for targeting a subgroup of the population to exclude them from using its services [4]. This highlights the fact that there is no mechanism to prevent them from misusing this power, for instance, to target a minority. Furthermore, the peer-to-peer car sharing platform Uber charges its customers based on an algorithmic prediction on how much they are willing to pay, rather than the services they receive [11]. Second, by keeping all data in one database, these platforms become attractive targets for attackers, resulting in several data breaches that violate the *privacy* of millions of users [2, 8]. Along with the fairness and privacy constraints, using these platforms entails an added cost to both, the user and the service provider. [16].

Fully decentralized solutions based on blockchain [13] are seen as a solution to the fairness issues of these centralized platforms. A blockchain is a trustless platform that allows any user to transfer funds and execute *smart contracts*–decentralized applications. Although the open nature of the blockchain provides better fairness guarantees, it is unclear how these solutions protect the users' privacy. Of the two blockchain-based asset-sharing schemes the authors are aware of, La'Zooz stores the private information in a centralized database [7], while the relevant code of Slock.it is closed-source, possibly under development [6]. Moreover, these systems provide a limited set of functionalities in comparison with their centralized competitors. La'Zooz only supports Consumer-to-Consumer (C2C) car-pooling in their advertisement, therefore it is unclear whether they can support C2C personal car sharing. As it appears, users of Slock.it cannot cancel a request once it is registered, therefore any misoperation might lead to financial loss for the user. Moreover, Slock.it has not yet been extended to car sharing. Robust car access provision protocols, such as SePCAR [14], are also limited in terms of functionality, as they do not provide a solution to booking and payments.

This work proposes a novel peer-to-peer car booking and payments system, named *SC*^{2}*Share*. SC^{2}Share works along the existing car access provision protocols such as SePCAR and uses a smart contract to register car sharing offers, match requests and settle payments. Compared with existing schemes, our system has the following advantages:

- **Fair.** Our system provides better fairness guarantees as it does not suffer from centralized price manipulation and membership exclusion.
- **Resistance against data breach.** There is no central point of failure in our design. Most interactions happen in a peer-to-peer fashion between the car owner and the consumer, which involves no intermediary. Sensitive information stored on the blockchain is encrypted with the receiver's public key, which is different for each participant. A compromised private key will not affect other users of the system.
- **Complete functionalities.** Our system offers complete functionalities that cover all typical execution paths in an Uber car sharing instance, including several conflict handling capabilities that ensure financial safety of each involved party against an adversary, the other party, and misoperation by the user himself. To the best of our knowledge, these combination of features is not offered in any existing decentralized car sharing platform.
- **Cost-effective.** As an additional advantage, our system is more cost effective to use due to the absence of a commission. The only cost involved is the deployment and transaction costs incurred by the blockchain. The experiments

with our deployed contract in Ethereum testnet highlights the operational costs and efficiency of SC2Share.

The remainder of this paper is organized as follows. Section 2 presents the necessary background. Section 3 presents the system model, threat model and design requirements used in our design. Section 4 describes SC^{2}Share, followed by its evaluation in Section 5. Finally, Section 6 concludes this paper and it offers future research directions.

2 Background

Since SC^{2}Share handles only the booking and payments aspect of car sharing, we first give a brief overview of SePCAR, a secure and privacy-enhancing car access provision protocol to generate and revoke car access tokens [14]. Then, we briefly introduce smart contracts in Ethereum - the main building block of SC^{2}Share.

2.1 SePCAR

SePCAR [14] is a car sharing scheme that offers strong security and privacy properties; it extends earlier work described in [15]. The system consists of various functional components. We only list the ones that are relevant to our system. An *Owner* is a user who is willing to share his car, a *consumer* is the user who wants to rent a car and *authorities* are the entities responsible for ensuring that the entire system is legal as well as for resolving any disputes between users.

The SePCAR protocol starts with the mutual agreement of booking details by the owner and consumer. Due to the presence of sensitive information such as the identities of the owner, consumer and the car along with its usage duration and location, the system server encrypts these booking details and an access token is generated. This access token is then stored in a public ledger, where it is retrieved by the consumer. Subsequently, a consumer uses this token to access the car without revealing his private information. SePCAR also guarantees the confidentiality of the car key and booking details.

2.2 Smart contracts in ethereum

Ethereum is a cryptocurrency with the third largest market capitalization [3]. Aiming at realizing a “world computer", Ethereum allows users to program smart contracts with a Turing-complete language and guarantees the correct execution of these contracts and the integrity of the system with its underlying blockchain [1].

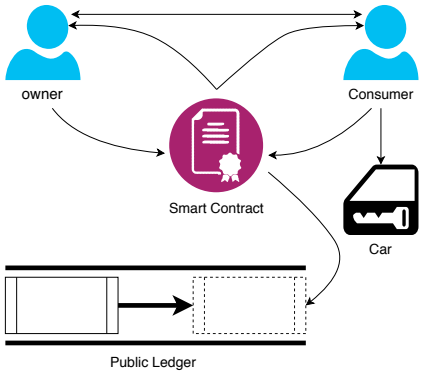


Figure 1.1: System model of SC2Share

Smart contracts are special accounts on Ethereum blockchain, that contain code and persistant storage along with an address and balance like normal accounts [10]. They are computerized protocols, that, without relying on any intermediaries, satisfy contractual conditions and minimize attacks by adversaries. As in any other computer program, the code of smart contracts also manipulates variables, and it can be invoked by sending a transaction to its address along with the required payment for its execution and parameters.

Miners, entities who embed transactions into the blockchain, are compensated by *transaction fees* in ether, the native currency of Ethereum, from the transaction initiators. The transaction fee is calculated as the total amount of *gas* consumed by the transaction execution, multiplied by the *gasPrice*, while the gas-ether exchange rate is specified in the transaction. The total gas is calculated by accumulating the gas consumption of all instructions of the execution. Each instruction has a predefined gas consumption. To prevent denial of service attacks, each transaction also specifies a *startGas*. A transaction execution that exceeds that startGas cannot be carried out, however, the transaction fee still goes to the miner.

3 System model, threat model and design requirements

This section introduces the system model, the threat model and assumptions as well as the functional, security and privacy requirements used in the design of SC2Share.

3.1 System model

As illustrated in Figure 1.1, our system model consists of the following entities:

- *Owner* - An individual with the intention to share his underused personal car.
- *Consumer* - A person in temporary need of a car.
- *Car* - The car that is available to be shared. Access provision to this car is handled by any robust access provision protocols such as SePCAR [14].
- *Public Ledger* - A structure that irrefutably records transactions and is operated by a single or several decentralized parties. All transactions made in SC2Share are recorded in this public ledger.
- *Smart Contract* - This smart contract is responsible for receiving a booking request, making sure that appropriate deposits have been made by both the owner and the consumer, handling cancellation requests, dealing with fraudulent activities, and a smooth rental procedure in general.

3.2 Threat model and assumptions

In SC^2Share the following threat model is used. The owner and the consumer are not trusted to carry out all transactions honestly. The integrity and correct execution of the smart contract is guaranteed by the underlying public ledger, in our case the Ethereum blockchain. Due to the blockchain's public nature, any private information embedded in the blockchain is considered leaked.

For our system to work, we also make the following assumptions. The owner and the consumer mutually agree upon the initial booking details and are successful in keeping them confidential against any third party. We also assume that the owner and the consumer have a public/private-key pair along with their digital certificates. Lastly, the communication channels are used by the owner and the consumer are private and authentic.

3.3 Design requirements

Our system should satisfy the following functional, security and privacy requirements.

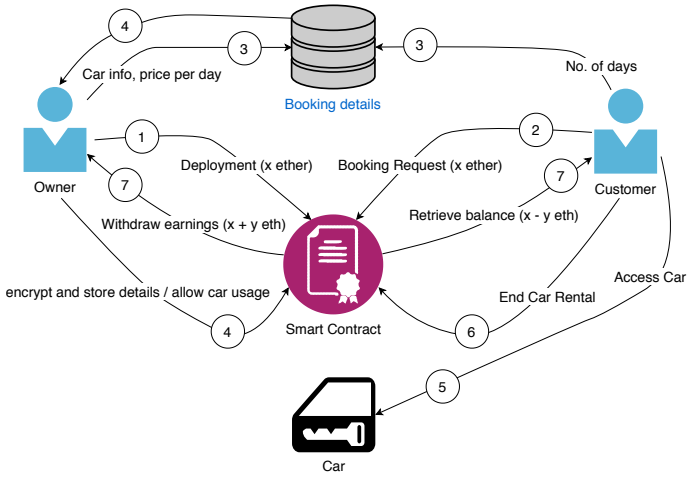


Figure 1.2: Overview of the SC2Share system

Functional requirements

- **Booking and payments process**: SC2Share should handle bookings, car usage and payments in an immutable way. It should be able to receive booking requests, safeguard deposits placed by owners/consumers and handle payments.
- **Fraud prevention**: SC2Share should be able to deal with fraudulent actions taken by both owners and consumers. Example actions are not making the car availlable and not returning the car. SC2Share should penalize the initiators of such fraudulent actions.
- **Cancellation**: SC2Share should handle cancellations of the booking agreement and distribute the deposit according to the booking agreement.
- **Extra time surcharge**: In case of extra time request by the consumer, SC2Share should be able to deduct the required extra amount from the consumer's deposit and transfer it to the owner.

Security and privacy requirements

- **Authenticity of booking details**: the integrity, origin and approval of the booking details should be validated by the car.

- **Non-repudiation of access token**: SC2Share should be able to prove the authenticity of the access token that is generated by the owner to the car and the consumer.
- **Forensic evidence provision**: In an adverse scenario of a fraudulent action, authorities should be able to retrieve forensic evidence from the system and gain access to specific booking details.
- **Confidentiality of booking details**: Only the owner and consumer should have access to the information stored in booking details. Booking details can hold personal identifiable and potentially sensitive information such as the license plate number, the location, booking time and type of the car.
- **Confidentiality of access token**: Only the car should have access to the information stored in the access token.
- **Anonymity of consumer and car**: The identity of the consumer and the car should be hidden to anyone except the owner, the consumer and the car.

4 SC2Share

This section provides an overview of how SC^{2}Share handles a car sharing scenario. Then, we explain in detail the algorithms that implement the functionalities described.

4.1 Overview of SC2Share

As it is illustrated in Figure 1.2, we provide a high-level overview of how our system works. It consists of seven steps described below.

1. An owner deploys a smart contract on the blockchain and deposits a value of x ether in it.
2. A consumer sends a request to rent the car by depositing x ether (the same amount as in step 1) in the deployed smart contract.
3. The owner and the consumer agree on mutually accepted booking details such as the required number of days to share the car, the pick-up and drop-off location and the daily price.
4. Once the details are agreed, the owner signs the booking details using his private key and encrypts them using the consumer's public key. Once encrypted, he sends these details to the storage of smart contract and allows the consumer

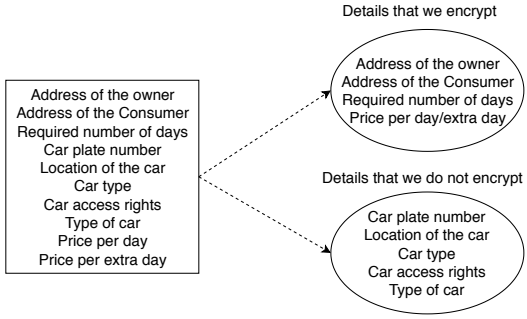


Figure 1.3: List of booking details

to access the car. The encrypted details stored in the smart contract can be accessed by the consumer and decrypted using his private key. However, as seen in Figure 1.3, we do not encrypt all information stored in booking details.

5. The consumer accesses the car and uses it for the agreed amount of time.

6. After completing the trip, the consumer ends the car rental process by dropping off the owner's car at the agreed location.

7. Once the process is over, both the owner and the consumer can separately withdraw their earnings and balances, respectively.

4.2 SC2Share in detail

This section describes SC2Share in detail. We first present the main protocol followed by the description of the cancellation, car access and extra time functionalities.

Algorithm 1 depicts a pseudo-code of how our system works. As explained in the overview, the system is initiated by the owner, which is followed by the consumer sending a request for booking. Once the consumer is set as the current driver of the car, the booking details are mutually agreed before being encrypted and stored on the blockchain. We use external libraries such as eth-crypto and eth-ecies to carry out the encryption off-chain, and to generate the identities of the owner and the consumer [9, 12]. The Javascript code that executes with the smart contract, and the smart contract itself are available online.

While storing the details in the smart contract, the owner has to set allow car usage value to 'true' in the smart contract to inform SC^{2}Share that the car can be used by the consumer. Once the consumer has access to the car, he can use it for the decided

Algorithm 1 SC^{2}Share main

```
1:  procedure SC2SHARE
2:  Owner Deploys Contract
3:      if Deployment successful then
4:          Customer Requests for Car Booking
5:          if Request successful then
6:              currentDriverAddress ← cAddress
7:              cDeposit ← x Ether
8:              oDeposit ← x Ether
9:              Mutually agree on booking details(BD)
10:             Encrypt BD and store
11:             if Owner allows car usage then
12:                 if Owner wants to cancel then
13:                     goto Algorithm 2;
14:                 if Customer access's car then
15:                     if Customer cancels then
16:                         goto Algorithm 2;
17:                     Use Car;
18:                     End Usage;
19:                     Owner withdraws earnings;
20:                     Consumer withdraws deposit;
21:                     if Extra time then
22:                         goto Algorithm 3;
23:                 else
24:                     if Customer cancels then
25:                         goto Algorithm 2;
26:             else
27:                 if Owner cancels then
28:                     goto Algorithm 2;
29:                 Customer withdraws total deposit;
30: Restart if deployment fails.
```

time. If they exceed the agreed time limit, the extra time surcharge is handled by Algorithm 4. In order to calculate extra time, the smart contract keeps deducting the mutually agreed 'price per extra hour' from the consumer's deposit. SC^{2}Share also has a fail-safe option, that is activated if the consumer fails to return the car. It awards the owner the total deposit stored in the smart contract, although it does not yet deal with the post-theft scenario. A possible solution to the post-theft scenario is provided by insurance businesses for sharing economy industry, such as SafeShare.

In cases where either the owner or the consumer wants to cancel the booking, Algorithm 2 comes into play. The pseudo-code explains how SC^{2}Share handles the cancellation at different stages of the system. If the owner cancels the booking before

allowing car usage, he is not penalized, but if they do it after allowing the car usage, then SC^{2}Share ensures that the consumer gets an incentive. The customer also pays the penalty based on whether they cancel the booking before accessing the car, or after.

Algorithm 2 Cancellation functionality

```
1:  procedure CANCELLATION
2:      currentDriverAddress ← cAddress
3:      cBalance ← consumer balance
4:      oBalance ← owner balance
5:      cDeposit ← consumer deposit
6:      oDeposit ← owner deposit
7:  Owner Allows Car Usage:
8:      if consumer access's car & cancels booking then
9:          x ← clientDeposit − penalty
10:         ownerBalance ← balance + x
11:         return true
12:     if owner cancels booking then
13:         x ← oDeposit − penalty
14:         cBalance ← cBalance + x
15:         return true
16:     if consumer cancels & does not access then
17:         return true
18: Owner Does Not Allow Usage:
19:     if owner cancels booking then
20:         oBalance ← oBalance + oDeposit
21:         cBalance ← cBalance + cDeposit
22:         return true
```

If the owner fails to give car access to the consumer after they have made the deposit, Algorithm 3 depicts how the owner is penalized for committing a fraudulent action as SC^{2}Share awards the total deposit stored in the smart contract to the consumer.

Once the consumer has successfully ended the car rental (see Algorithm 1), SC^{2}Share calculates the amount that has to be deducted from their deposit and adds it to the owner's deposit, after which the owner and consumer can separately withdraw their money.

5 Evaluation

In this section, we analyze our design and provide a description of how SC2Share fulfills the functional, security and privacy requirements. We also provide the real-world deployment cost of SC^{2}Share on the Ethereum blockchain along with all the transaction costs it incurs while being used. Evaluation on the basis of total time taken

Algorithm 3 Car access functionality

```
1:  procedure CAR ACCESS
2:      currentDriverAddress ← cAddress
3:      cBalance ← consumer balance
4:      if owner allows usage then
5:          carStatus ← ready
6:          consumer ← access
7:          return true
8:      if owner does not allow usage then
9:          carStatus ← notready
10:         consumer ← noAccess
11:         consumer can retrieve total deposit
12:         x ← totalDeposit
13:         cBalance ← cBalance + x
14:         return true
```

is tricky as the execution time for each transaction in SC^{2}Share can only be aggregated since there is no concrete way of knowing how long a transaction would take before its included in Ethereum. As of January 2019, the median wait time between each block is ≈ 32 seconds.

5.1 Functionality analysis

SC^{2}Share uses the Ethereum blockchain to implement the booking and payment functionalities for car sharing applications. All transactions related to car rental, are immutable when published on the blockchain. Once a transaction is made using SC^{2}Share, the owner or the consumer cannot refute the ownership of it. Moreover, no outsider knows the identities of the owner and consumer, and all the sensitive information in booking details is encrypted before being stored on-chain.

The Algorithm 3 and 4 demonstrates how to mitigate a malicious owner or consumer and hence prevent *fraudelent actions*. Possible fraud behaviour is countered with a penalty of losing the deposit made in the smart contract. The Algorithm 4 also enables the functionality of *extra time surcharge*.

Moreover, in SC2Share we consider all possible cancellation scenarios, and a fair settlement for both the owner and consumer in Algorithm 2. Hence we guarantee the functionality of *cancellation*.

Algorithm 4 Extra time functionality

```
1:  procedure EXTRA TIME
2:      currentDriverAddress ← cAddress
3:      cBalance ← consumer balance
4:      oBalance ← owner balance
5:      cDeposit ← consumer deposit
6:      oDeposit ← owner deposit
7:  consumer Ends Car Rental:
8:      if end time =< decided time then
9:          currentDriverAddress ← null
10:         carStatus ← idle
11:         x ← cDeposit − rentCharge
12:         oBalance ← oDeposit + x
13:         cBalance ← cDeposit
14:         return true
15:     if end time > decided time then
16:         currentDriverAddress ← null
17:         carStatus ← idle
18:         x ← cDeposit − rentCharge
19:         y ← cDeposit − chargePerExtraHour
20:         z ← x + y
21:         oBalance ← oDeposit + z
22:         cBalance ← cDeposit
23:         return true
24: Consumer Runs Away With Car:
25:     x ← totalDeposit
26:     oBalance ← oBalance + x
27:     return inform insurance
```

Total Size (bytes)	Deployment Cost in gas	Deployment Cost in USD
8520	1,352,283 gas	$0.21623

Table 1.1: Deployment costs of SC2Share.

5.2 Security and privacy analysis

Before publishing the booking details in the blockchain, the owner has to encrypt and sign them. SC^{2}Share verifies the owner's signature on the booking details before giving access to the consumer to verify if the owner also agrees with the conditions set for the car rental, thus providing *authenticity of booking details*.

Car access provision protocols such as SePCAR [14] use the encrypted booking details to generate the access token. SC^{2}Share verifies the origin of encrypted details in

Transaction	Cost in gas	Cost in USD
set required days (customer)	42,131 gas	$0.00674
store encrypted details in contract (owner)	63,022 gas	$0.02739
rent car (customer)	112,220 gas	$0.01795
allow use of the car (owner)	28,655 gas	$0.00459
access the car (customer)	29,038 gas	$0.00464
cancellation of booking (owner/customer)	76,342 gas	$0.0122
ending the rental procedure (owner/customer)	82,590 gas	$0.01321
triggering the distribution of earnings (owner)	32,992 gas	$0.00528
withdrawal of money (owner/consumer)	22,099 gas	$0.00353

Table 1.2: Costs of executing each transaction in SC2Share.

order to make sure that no one but the owner has signed them. This is achieved by comparing the address of the signer to the address of the owner. This is how we ensure *non-repudiation of access token*.

Smart contracts are auditable by nature, and in case of an incident where the owner or the consumer is involved, the SC^{2}Share can be audited to reveal private information about the owner or the consumer guaranteeing the requirement of *forensic evidence provision*.

SC^{2}Share ensures the *confidentiality of booking details* by enabling only the owner and the consumer to have access to the sensitive information stored in booking details. These details are discussed offline using a secure channel (see assumptions in Section 3) and only stored in the smart contract when encrypted.

SC^{2}Share treats the encrypted booking details as the access token and stores them in its internal storage. No one except the consumer and the owner can decipher an access token thus guaranteeing *confidentiality of the access token*.

Anonymity of consumer and car is provided by design in SC^{2}Share. On the blockchain, the consumer can choose to be anonymous by using his 20-byte address to interact with the contract and not publicizing any personal details, while the identity of the car is encrypted along with the other private information by the owner before storing it on-chain.

5.3 Deployment and usage cost

The deployment of a smart contract and the interaction with it has a cost for the caller of the transaction, and this cost is calculated in gas. This section discusses the deployment

and interaction costs associated with SC^{2}Share. It is important to mention that the price of ether per gas unit is volatile, and these results are obtained by using the price of ether per gas unit for average confirmation time as of January 2019. To evaluate the total cost of SC^{2}Share, we look at two different costs, the *deployment* and the *transaction cost*.

Deployment cost

The main costs for deploying a smart contract can be calculated by combining the following parameters [17]:

- The costs associated with storing the contract code (200 gas per byte);
- Additional data storage cost on the contract (20,000 gas per 256-bit word);
- Each transaction has a base cost of 21,000 gas, and 32,000 gas have to be paid for a CREATE transaction (deployment of a new contract);

Ganache was used as a provider for web3.js to perform tests on SC^{2}Share and calculate the estimated gas for deployment, as well as the execution of each transaction. The absolute deployment cost is influenced by the size of the contract and the total bytes in its storage. Table 1.1 illustrates the deployment cost of SC^{2}Share in USD. It can be calculated by multiplying the total gas with the cost per gas unit in USD.

Transaction cost

The cost for executing each transaction can be calculated as follows [17]:

- The base cost of each transaction (21,000 gas);
- The cost for storing a 256-bit word on the smart contract (20,000 gas);
- The cost for editing data stored in the smart contract (5,000 gas);
- The cost of making a transaction call having a monetary value (9,000 gas);

Table 1.2 illustrates the cost of executing each transaction in SC^{2}Share. Adding up the cost of all individual transactions, the total cost of all transactions in SC^{2}Share is USD 0.095. Hence, it can be deduced that our smart contract implementation is not expensive to deploy and operate.

6 Conclusions and future work

This paper presented a fully decentralized car booking and payments system known as SC^{2}Share. This system can be incorporated with car access provision protocols to provide a secure and private car sharing environment without the need of any intermediary. We have shown that SC^{2}Share provides all major functionalities that are required for a car sharing platform, and provides security and privacy by design. The total cost of deploying and using our system on the Ethereum network sums up to USD 0.312, which in comparison to the commission fee paid to large organizations is relatively cheap. Hence, we conclude that along with being functionally sound, secure and private, SC^{2}Share is also cost effective for its users.

As future work we would like to advance our system design and implementation to work with fully encrypted booking details, including the price per day, price per extra day and required number of days which are used for calculation of payments, as well as to provide formal security and privacy proofs of the advanced system.

Another potential direction could be to adapt SC2Share so that it supports the use of advanced cryptographic primitives such as zero-knowledge proofs. Ethereum Whisper could also be used to possibly remove the waiting time of customer to get car access.

References

[1] Vitalik Buterin. A next-generation smart contract and decentralized application platform. URL: http://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf, last checked on 2018-12-20, 2014.

[2] Paul Carsten. Hackers attack 20 mln accounts on alibaba's taobao shopping site. URL: https://www.reuters.com/article/alibaba-cyber-idUSL3N15J1P2, last checked on 2018-12-10, 2016.

[3] CoinMarketCap. Top 100 cryptocurrencies by market capitalization. URL: https://coinmarketcap.com/, last checked on 2018-12-20, 2018.

[4] Keith Collins. A running list of websites and apps that have banned, blocked, deleted, and otherwise dropped white supremacists. https://qz.com/1055141/what-websites-and-apps-have-banned-neo-nazis-and-white-supremacists/, last checked on 2023-05-08, 2018.

[5] Michael A. Cusumano. The sharing economy meets reality. *Commun. ACM*, 61(1):26–28, December 2017.

[6] Christopher Jentzsch. Decentralized autonomous organization to automate governance. URL: https://download.slock.it/public/DAO/WhitePaper.pdf, last checked on 2018-12-10, 2016.

[7] La'Zooz. La'zooz white paper. URL: https://www.weusecoins.com/assets/pdf/library/LaZooz%20Blockchain%20Taxi%20Whitepaper.pdf, last checked on 2018-12-10, 2015.

[8] Dave Lee. Uber concealed huge data breach. URL: https://www.bbc.com/news/technology-42075306, last checked on 2018-12-10, 2017.

[9] libertylocked. eth-ecies. URL: https://github.com/libertylocked/eth-ecies, last checked on 2018-12-20, 2018.

[10] Loi Luu, Yaron Velner, Jason Teutsch, and Prateek Saxena. Smart pool: Practical decentralized pooled mining. *In USENIX Security Symposium*, 2017.

[11] Eric Newcomer. Uber starts charging what it thinks you're willing to pay. URL: https://www.bloomberg.com/news/articles/2017-05-19/uber-s-future-may-rely-on-predicting-how-much-you-re-willing-to-pay, last checked on 2023-05-08, 2017.

[12] pubkey. eth-crypto. URL: https://github.com/pubkey/eth-crypto, last checked on 2018-12-20, 2018.

[13] Toshendra Kumar Sharma. What does blockchain means for a sharing economy? URL: https://www.blockchain-council.org/blockchain/what-does-blockchain-mean-for-a-sharing-economy/, last checked on 2018-12-10, 2018.

[14] Iraklis Symeonidis, Abdelrahaman Aly, Mustafa A. Mustafa, Bart Mennink, Siemen Dhooghe, and Bart Preneel. SePCAR: A secure and privacy-enhancing protocol for car access provision. *in the 22nd European Symposium on Research in Computer Security (ESORICS'17), ser. LNCS, vol. 10493. Springer, 2017*, pages 475–493, 2017.

[15] Iraklis Symeonidis, Mustafa A. Mustafa, and Bart Preneel. Keyless car sharing system: A security and privacy analysis. *In IEEE International Smart Cities Conference (ISC2 2016)*, pages 1–8, 2016.

[16] Pascal Thellmann. Blockchain is the infrastructure for a new decentralized sharing economy. URL:https://hackernoon.com/blockchain-is-the-infrastructure-for-a-new-decentralized-sharing-economy-f715da32bece, last checked on 2018-12-5, 2018.

[17] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger Byzantium version. https://ethereum.github.io/yellowpaper/paper.pdf, last checked on 2018-08-5, 2014.

Publication 2

Exploring the Storj Network: a Security Analysis

Publication Data

Sammy de Figueiredo, Akash Madhusudan, Vincent Reniers, Svetla Nikova, and Bart Preneel. "Exploring the storj network: a security analysis". In Chih-Cheng Hung, Jiman Hong, Alessio Bechini, and Eunjee Song, editors, SAC '21: The 36th ACM/SIGAPP Symposium on Applied Computing, Virtual Event, Republic of Korea, March 22-26, 202, pages 257–264. ACM, 2021

Contributions

- I was a co-author involved in Conceptualization, Methodology, Writing - Review & Editing, and Resources.

Exploring the Storj Network: a Security Analysis

Sammy de Figueiredo, Akash Madhusudan, Vincent Reniers, Svetla Nikova, and Bart Preneel

KU Leuven, ESAT/COSIC and KU Leuven, Distrinet, and imec, Leuven, Belgium
{akash.madhusudan, svetla.nikova, bart.preneel}@esat.kuleuven.be
{vincent.reniers}@cs.kuleuven.be
{sammy_de_figueiredo}@hotmail.com

Abstract. The recent decade has seen a tremendous increase in volumes of data consumed and generated. Towards storing such data, companies typically make use of centralised cloud storage systems which allow for on-demand scalability and a pay-per-use model.

These storage providers, while presenting many benefits, have several downsides in terms of (i) posing as a single point of failure (e.g. data breaches), (ii) featuring the potential for misuse of confidential or personal data, as well as (iii) being not nearly as competitive as they could be which has resulted in inflated prices.

In recent years, along with blockchain-based technologies, novel distributed storage platforms such as Storj and Sia, have emerged that tackle these issues by enabling a dynamic storage market between hosts and renters. In comparison to traditional Peer-to-Peer (P2P) storage solutions (e.g. BitTorrrent), these platforms rely on incentivization-based P2P hosting which ensures file availability, robustness and price competitiveness, while also preserving confidentiality and ownership through end-to-end encryption.

In this paper, we present a brief overview of such storage platforms, as well as provide two major contributions in the form of (i) an architectural overview, as well as (ii) a high-level security exploration on the dev./test environment of Storj. The latter has resulted in the discovery of a DoS vulnerability, which we experimentally evaluated to be highly feasible. We have contacted Storj and they assessed that their production system[1] is not vulnerable to such an attack.

[1] Multiple satellites running on load balanced clusters of servers.

1 Introduction

Nowadays behemoths like Amazon, Google and Microsoft abstract away the complexity of cloud storage and store our data in large data centers.

The centralized architecture used by these client-server cloud providers introduces a number of problems. Firstly, because a single point of failure/attack exists, it makes them more vulnerable to data breaches [23][15], data outages [1][35] and facilitates censorship [10]. Secondly, a lack of privacy and data ownership allows big corporations to misuse our personal data [41] in a way that best befits their for-profit strategy [6]. Thirdly, as the cloud storage market is not nearly as competitive as it could be, prices can be kept artificially high [28][34][27].

Numerous initiatives have been undertaken to develop (partly) decentralized storage solutions which no longer suffer from the above mentioned problems. This in term has given rise to open source projects like Sia, Storj, Swarm, Filecoin (+ IPFS), MaidSafe and many others. Compared to traditional client-server cloud providers, these systems are characterized by peer-to-peer (P2P) architectures that significantly lower infrastructural requirements. This is one of the main reasons they are able to provide the same (or even higher) levels of data availability and durability at lower prices. Most of them guarantee data ownership and privacy by encrypting any data before it leaves the customer's device.

To the best of our knowledge, little to no research has yet fully analysed and compared these platforms on either (a) architectural characteristics, (b) functionality, and (c) security. The latter of which is the main focus due to the potential issues such as data loss, privacy and confidentiality, that can arise when improper care is taken.

Our work addresses this scarcity by examining one of the more prominent platforms, more precisely Storj, which was deemed a good candidate for closer inspection due to its production readiness, popularity and its extensive documentation. This paper presents the following contributions :

1. An architectural overview of Storj's decentralized incentivization-based file storage network.
2. A DoS attack on Storj's dev./test environment which can result in a network downtime, leading to potential data and monetary loss.

The rest of the paper is organized as follows: Section 2 provides a background on incentivization-driven distributed storage platforms. Section 3 describes the architecture of a mature storage platform, namely Storj. The security evaluation in Section 4 provides a detailed description of the weakness in Storj's architecture which led to

Table 2.1: Incentivization-driven distributed storage systems.

Storage platform	Architecture	Incentivization	Functionality	Status network
Sia [37]	Decentralized	Cryptocurrency	Smart contracts	Live
Storj [18]	Distributed	Bookkeeping centralized service	Audit, repair, ...	Live
Swarm [32]	Decentralized	Cryptocurrency	Smart contracts	Alpha
Filecoin [13]	Decentralized	Cryptocurrency	Smart contracts	Test

this vulnerability together with the setup that was used. Section 5 discusses potential solutions to our attack and finally in Section 6 we conclude our paper and list future research directions.

2 Incentivization-driven distributed file storage systems

Peer-to-peer (P2P) storage solutions have become widely adopted ever since their first introduction in platforms such as Napster in 1999, which enabled users to exchange files via a centralized index [31]. The first decentralized file sharing network emerged with Gnutella in March 2000 [5], which was in widespread use by client applications such as Limewire [20] and Kazaa [21]. However, availability of the files hosted on these P2P networks is only warranted by numerous hosts actively sharing them.

In recent years, the emergence of P2P blockchain-based networks, and cryptocurrencies, has allowed to tackle the file availability problem through monetary incentivization.

2.1 Overview

Numerous such incentivization-driven distributed file storage systems have emerged over the last several years, such as Storj [18], Sia [37], Filecoin [13] and Swarm [32]. In addition to guaranteeing file availability via compensation of storage hosts, and even automated file repair, these solutions can also solve the problem of data confidentiality and privacy by encrypting the data before it gets distributed on the network.

Table 2.1 depicts an overview and characterizes existing storage technologies on the basis of their storage architecture and manner in which storage hosts are compensated.

Storj chooses to trade-off security for speed by using a partly centralized architecture. In contrast, the other solutions use a fully decentralized architecture combined with

blockchain technology [2], thereby solving the single point of failure/attack problem faced by traditional cloud storage providers without jeopardizing security.

All of the blockchain based solutions employ some variation of a smart contract. Sia and Filecoin, for example, have adapted the transactions described within the bitcoin paper [22] to their own needs. On the other hand, the Swarm storage platform is deeply integrated with the Ethereum [12] blockchain and its corresponding smart contracts.

With respect to privacy and data ownership, both Storj and Sia encrypt any data before it leaves the client node. Alternatively, Filecoin does not provide such functionality out of the box. Swarm tries to find a balance between extremes by supporting encryption without imposing it to their customers.

Depending on the SLA, contractual relations among network participants vary from system to system. Storj, for example, enforces hard limits[30] consisting of at least 2 TB of bandwidth available per month, 5 Mbps upstream, 25 Mbps downstream and a minimal uptime of 99.3% per month. On the other hand Sia embraces a more flexible approach where hosts publish their contracts(pricing, uptime, etc.) and renters can agree to these terms.

2.2 Related work

Janhvi and Rejo [17] have presented a survey of attacks that can be executed on the Bitcoin network. Since some of these novel storage solutions like Sia derived their blockchain model from Bitcoin, similar exploits might be applicable to them. Similarly, a vulnerability in the Ethereum platform [7] could in turn open the door for a variety of attacks on Swarm since the latter is deeply integrated with the former.

More fundamental research by Zhang and Preneel [39] also showed that the proof-of-work (PoW) protocol is unable to withstand certain attacks like selfish mining, double-spending and feather-forking. We expect these vulnerabilities to be equally exploitable in systems like Sia that employ such a protocol.

In addition to these related attacks, specific ones do exist but are rather few in number. Zhang et al. [40], for instance, demonstrated how to frame an innocent victim in the Storj network by storing unencrypted data on their hard drive. This attack, however, applies to an earlier version of Storj which is no longer running, demonstrating the dedication of the Storj team to ensure their system is secure. A computer security research firm, called RET2SYSTEMS, disclosed an issue [33] in Sia that allowed both coin theft and remote code execution. However, the issue was addressed very quickly thus preventing any exploitation.

Since Sybil attacks [9] can only be mitigated in P2P networks, the same applies for Eclipse attacks as the latter depends on the former. An example of such an attack,

targeting the Bitcoin network, was presented by Heilman et al. [16]. As all of the novel open source solutions are P2P networks, one needs to pay attention to these security aspects while evaluating these platforms.

Vokerla et al. [36] provided an overview of attacks that can be launched by hackers and scammers in blockchain contexts. A comprehensive survey on the blockchain-based storage systems and how they work as compared to cloud-based storage networks can be found in Benisi et al. [3]. Reniers et al. [26] provides a comprehensive overview of architectural variants for decentralized control of private encrypted data, and the involved trade-offs in terms of performance, storage overhead, auditable trust and security.

3 Storj architecture

We start off by describing the general architecture of one of the most prominent P2P file storage network, namely Storj. In Section 3.1, we will first provide a high level overview of the system and describe its major services. Then, in section 3.2, we will zoom in on the dynamic behavior of the system by analyzing a set of operations that will expose us to more details.

3.1 Services

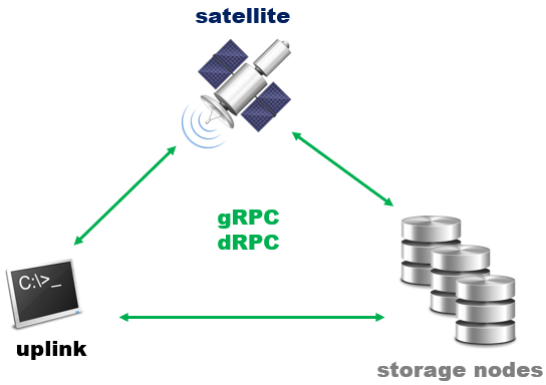


Figure 2.1: Network level view of Storj

In figure 2.1, we can discern three major services that can communicate via both gRPC [14] and dRPC[2] [29] :

[2]A light replacement for gRPC, developed by Storj.

- Storage node - A storage node tries to earn money by leasing excess hard drive space to customers. In order to achieve this, it first assigns a certain amount of disk space and bandwidth to one or more satellites. These settings together with a STORJ token[3] wallet address are subsequently sent to the satellites.
  The task of a storage node consists of both returning data when requested and storing data at rest. Payments made to a storage node, in the form of STORJ token, are directly related to how well that task is performed.
- Uplink - An uplink provides a customer with a command-line interface that enables him to perform a variety of tasks related to storing and retrieving data. Encryption, erasure encoding and coordination with the other two peer classes are all handled by the uplink.
- Satellite - Most of the systems complexity is managed by the satellite. Some of its duties include storing metadata, paying storage nodes, performing audits and repairs. Note that in contrast to the other parts, a satellite must be fully operational at all times for the system to work correctly. This single point of failure renders the whole system more vulnerable to attacks.

Although any subset of these three services could be hosted on the same node, they are typically hosted on separate nodes.

In theory, any node is eligible to perform the role of a satellite. However, this results in trust issues since no guarantees can be given with respect to its integrity. Hence, currently the satellites are exclusively hosted by Storj Labs.

3.2 System behavior

This section describes critical functionality within the Storj architecture. Section 3.2 describes the registration process, whereas section 3.2 describes the upload operation. Finally, in sections 3.2 and 3.2 we respectively take a look at the audit and repair functionalities.

Customer registration process

Before being able to interact with the Storj network a customer must select a satellite and create the following items via the Tardigrade [19] web interface:

1. An account to access the satellite.

[3] An ERC-20 token running on the Ethereum platform.

2. A project that allows the customer to invite team members, manage billing and access to various apps or users.

3. An API key employed by the uplink to connect to the satellite.

After completing the above mentioned steps, the uplink will be used to set up a bucket on the selected satellite. A bucket is just a way to group identically encrypted/encoded data. This process is displayed in figure 2.2.

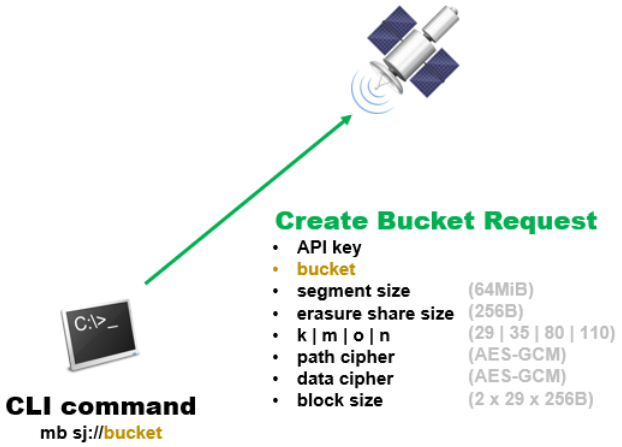


Figure 2.2: Creating a bucket on a satellite

Upon receiving the request, the satellite first checks whether the corresponding action is entitled to execution or not. This is achieved by verifying the authorization level associated with the provided API key[4]. An API key forms the link that uniquely ties a customer to a specific satellite. Note that the previously mentioned authorization step is performed for every request presented to the satellite.

The remaining parameters of the request determine the encryption and erasure settings related to the bucket. Storj added two extra values to the standard Reed-Solomon [25] tuple (k,n). The value m and o respectively determine when repair is triggered and redundant uploads/repairs are canceled. As illustrated in figure 2.3, this creates a safety buffer of $m - k$ nodes for repair and makes the system immune to the $n - o$ slowest nodes.

[4] A macaroon[4] to be precise

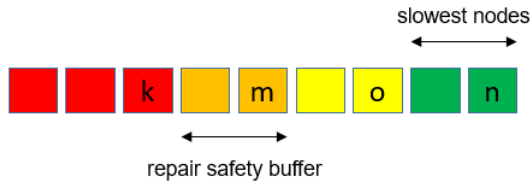


Figure 2.3: The Storj erasure numbers

File Upload

An upload procedure starts by retrieving the default settings of the selected bucket. Some of the settings are overwritten by the satellite before being used by the uplink to encrypt and erasure encode the chosen file.

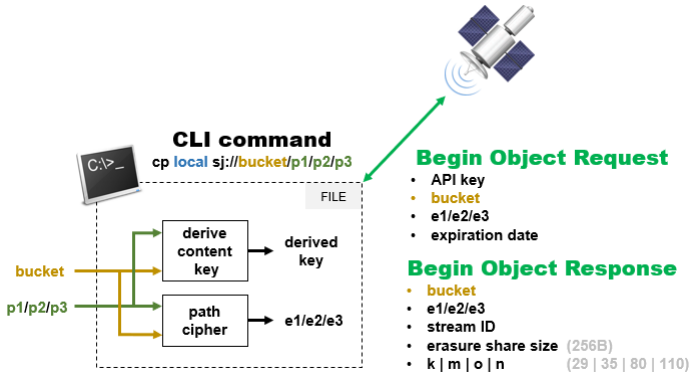


Figure 2.4: Initial upload of encrypted path and optional expiration date

As depicted in figure 2.4, the next step consists of encrypting the Storj network destination path as well as generating a derived key. The API key, bucket name, encrypted path and expiration date are then sent to the satellite. The expiration date determines the point in time after which storage nodes are no longer obligated to preserve the uploaded file. Indefinite storage is assumed by default in case no expiration date was specified. The satellite responds to the request by returning the bucket name, encrypted path, final erasure parameters and stream ID. The subsequent step involves streaming the file and encrypting it on a segment to segment basis.

Segments smaller or equal than the inline threshold[5] do not require to be erasure encoded as they are directly stored on the satellite. Figure 2.5 illustrates such a

[5] 4kiB by default

situation where the inline segment stream gets encrypted by a content key K_c and content nonce N_c.

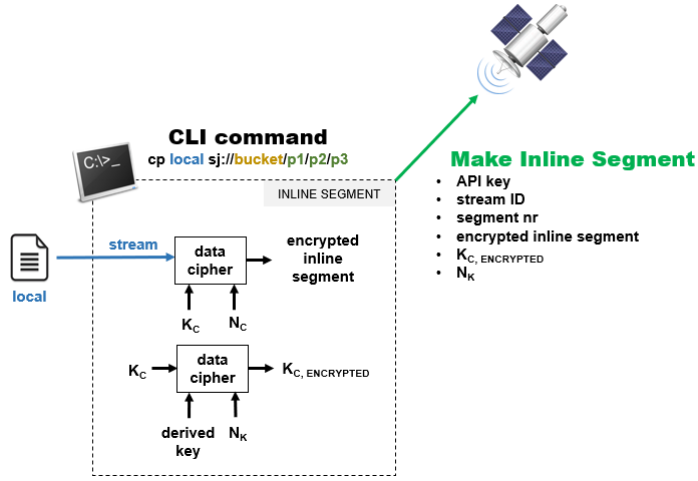


Figure 2.5: Uploading an inline segment to the satellite

The initial value of the content nonce is set to one and incremented with every segment. The uplink also encrypts the content key with the previously derived key and a key nonce N_K. Both the encrypted inline segment and content key are transmitted to the satellite alongside with the key nonce, segment number and stream ID.

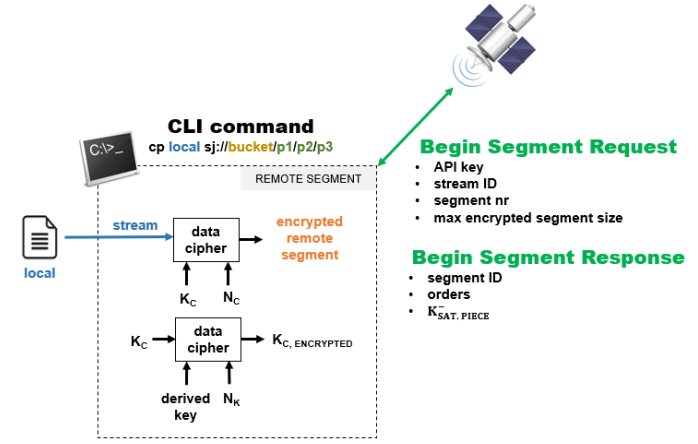


Figure 2.6: Contact satellite before uploading a remote segment

A segment that exceeds the inline threshold will be stored on a set of storage nodes. To achieve this, the uplink first encrypts the remote segment and content key in exactly the same way as with the inline segment. As depicted in figure 2.6, it then sends a request containing the maximum encrypted segment size to the satellite.
Based upon that value, the satellite reacts to the request by searching for a set of suitable storage nodes. After the set is found, the satellite generates a root piece ID and an asymmetric piece key pair $(K^{-}_{SAT,PIECE}, K^{+}_{SAT,PIECE})$. Additionally, an order gets created and signed by the satellite for every storage node within the set. The satellite finally responds with the private piece key $K^{-}_{SAT,PIECE}$, a segment ID and the list of orders.

After having received those values, the uplink is now able to process the remote segment which it first divides into stripes. Each stripe is erasure encoded into shares as shown in figure 2.7. Groups of shares with the same index, called pieces, are then formed. The uplink finally sends each piece to a different storage node.

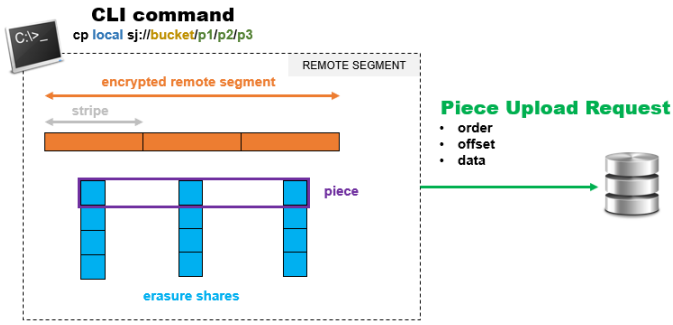


Figure 2.7: Uploading one piece to a storage node

Remark that the piece upload request includes one of the orders that was generated by the satellite. Both the uplink and storage node use order fields to respectively upload data and claim satellite payment.

After having uploaded a piece, the uplink transmits the following upload result items after having signed them with the private satellite piece key $K^{-}_{SAT,PIECE}$:

- Piece ID
- Piece size
- Hash
- Timestamp

The storage node verifies the authenticity and integrity of those fields by utilizing the satellites public piece key $K^{+}_{SAT,PIECE}$ included in the order. It then reproduces the same step with the difference that the piece size and hash are based upon the received data. Furthermore, signing is now done by applying the private key of the storage node. The final move of the storage node consists of sending the fields to the uplink who in turn verifies for authenticity and integrity as well.

When at least o pieces have completed their upload successfully, the uplink gathers all the metadata associated with the remote segment. This includes the segment ID and size, its key nonce N_K and encrypted content key $K_{c,\,\text{ENCRYPTED}}$ as well as the results from the piece upload requests. This situation has been illustrated in figure 2.8. This final message concludes the upload procedure.

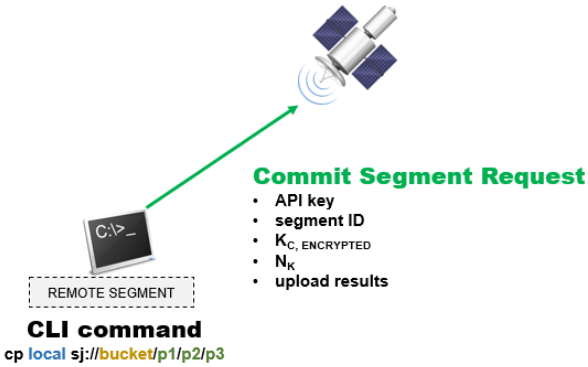


Figure 2.8: Sending metadata of a remote segment to the satellite

File Audit

A satellite organises audits for the storage nodes falling under its management. To accomplish this, the satellite goes through all metadata of the remote segments while picking at most three segments per storage node. Next, those remote segments are enqueued so that they can be processed at their own pace. When processed, a stripe is randomly chosen from the segment. Then its corresponding erasure shares are requested by the satellite from the storage nodes that are expected to faithfully safeguard them. Such a situation has been depicted in figure 2.9.
In the best case scenario, all storage nodes would be able to return their shares correctly. This requires no additional work from the satellite.

If however, a storage node does not respond within a generous time frame then it will be placed in containment. As a result, the satellite will continue to query the same erasure share from the storage node until it responds or it is disqualified. Answering

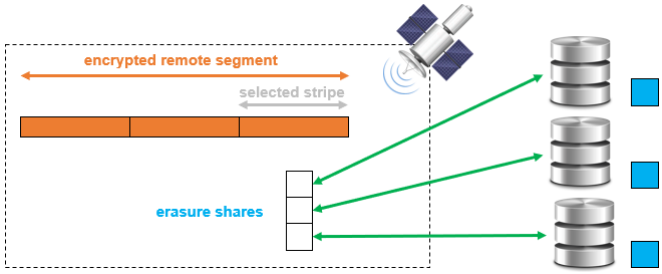


Figure 2.9: The auditing process scoped to one stripe

incorrectly or being disqualified results into a diminution of payment. In the worst case, this can even result in no payment at all and exclusion from the set of potential storage nodes.
In order to correct erroneous shares and determine which storage nodes are to be penalized, the satellite runs the Berlekamp-Welch error correction algorithm [38].

File Repair

In case the amount of properly returned erasure shares falls below a certain threshold m during an audit, its corresponding remote segment will be included into the repair process.
This process first downloads a sufficient amount of pieces to enable the correct reconstruction of the remote segment. Next, the satellite selects a set of storage nodes to which new pieces will be uploaded. As a final step, the metadata of the remote segment in question is updated.

4 Security evaluation

After the architectural description, we now focus on the security aspect of the system. We will first clarify how the vulnerability works in Section 4.1 before describing the experimental context that was used to exploit it in Section 4.2.

4.1 DoS vulnerability in Storj's dev./test environment

Section 3.1 shows that the satellite plays a key role. Its failure would render the whole system useless as users would no longer be able to access their data. This single point of failure within the system makes a dev./test satellite an ideal candidate for a (D)DoS

attack. With respect to the vulnerability of a satellite in the production environment, this is an untested claim due to the scope our experiment, i.e, Storj's test network.

In order to evaluate whether a satellite is able to withstand (D)DoS, our initial intent was thus to deploy it on the Storj system with the help of the network stress tool LOIC [24]. However, the problem with this approach was that the tool did not support TLS functionality out of the box. Furthermore, the data was sent in a format which did not match the one used by the satellite.

Instead, we have adapted the Storj code [8] itself to stress test the network, which does not require any adaptation of the data format, and the connections are already present. We then chose a point in the code base were a connection was established between the satellite and another component. In our case we chose a storage node, as depicted in figure 2.10 below, but we would like to point out that the uplink would also been valid candidate.

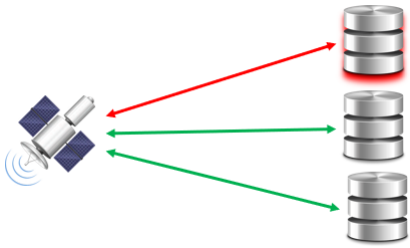


Figure 2.10: A DoS attack orchestrated by one malicious storage node

At first, we tried to call the existing 'pingSatelliteOnce' function in an infinite loop without any delay. This clearly deviates from the behaviour a regular storage node would exhibit, i.e., calling this function once every 30 seconds. Besides a small increase in the usage of the satellites resources, no adversarial results could be noticed by using the method outlined above.

While conducting the experiments we nevertheless noticed that closing the connection within the 'pingSatelliteOnce' function took some time. Therefore, the next logical step was not to close any connection within our infinite loop. As a consequence, we would be able to establish more connections in shorter amount of time. The satellite would in term use more resources which then could result in a DoS. In case a single storage node turned out to be insufficient to cripple the satellite, more malicious storage nodes could be added consequently scaling the attack to a DDos.

However, not closing the connections resulted in a complete satellite crash. After carefully investigating the code, we were able to pinpoint the reason for the crash. Every time a connection gets established via dRPC[29] by a storage node, the satellite spins up the number of go routines. These go routines are terminated after receiving a

'close' message from the storage node or as a consequence of a TCP connection timeout on the server side, in the kernel. Before such a timeout is reached, the management of the satellites resources is thus directly linked to the storage nodes' behaviour. This results in a steady growth of go routines that consume the satellites resources if connections are not correctly closed. When the resources consumed exceeds a certain limit, the satellite can no longer function correctly and crashes. In our test environment setup on a Win10 machine this limit lies on the 8192 mark.

A sample of our malicious code is presented below in listing 2.1. It is located in the function 'pingSatellite' within the file 'storj/storagenode/contact/service.go'. In essence, it is an exact copy of the 'pingSatelliteOnce' function with the omission of the connection closure.

```
if test {
    address, er := service.trust.GetAddress(ctx, satellite)
    if er != nil {
      return errPingSatellite.Wrap(er)
    }
    conn, er := service.dialer.DialAddressID(ctx, address, satellite)
    if er != nil {
      return errPingSatellite.Wrap(er)
    }
    self := service.Local()
    _, er = pb.NewDRPCNodeClient(conn).CheckIn(ctx, &pb.CheckInRequest{
      Address:  self.Address.GetAddress(),
      Version:  &self.Version,
      Capacity: &self.Capacity,
      Operator: &self.Operator,
    })
    if er != nil {
      return errPingSatellite.Wrap(er)
    }
    continue
}
```

Listing 2.1: Malicious code sample

Its behaviour now depends upon the boolean 'test' parameter in the config file. A regular node will continue to establish a connection with a satellite at 30s[6] intervals to notify the latter that it is still available. A malicious node, on the other hand, will establish as many connections as possible without ever closing one.

We would like to point out that above situation which involves only one satellite could be generalized to multiple satellites. In a worst case scenario this would mean that a single malicious storage node could connect to all the satellites of which the Storj system is comprised. Subsequently executing the attack in a synchronized manner would effectively bring down the whole system.

[6]This is the default value.

Finally we want to underline the fact that Storj uses the dRPC library to communicate in various ways between its components. As a consequence, all parts of the system that make use of the dRPC library to establish a connection are vulnerable to our attack.

4.2 Experimental security evaluation

As exploiting potential vulnerabilities on a live network would be unethical, our experiments have been conducted on the local Storj dev./test environment. In such a setup, the three major parts can run on a single node.

Our setup consists of a single satellite and ten storage nodes. Although registration to multiple satellites by a single customer could provide a redundant backup, we do not expect that to be the general use case for Storj as it would multiply the original storage costs by the number of satellites used.

Setting up such an environment is quite straightforward as the source code already includes a command to spin up a number of satellites and storage nodes. The supplementary steps needed in the process are described on Github [11].
Some caution must nonetheless be exercised when creating the Docker container that contains the PostgreSQL database. Listed below are the four commands that configure and run a local Storj network :

1. docker run
   --name <name>
   -e POSTGRES_PASSWORD=<pwd>
   -p 5432:5432
   -d postgres
2. docker exec -it postgres createdb -U postgres <db>
3. storj-sim network setup
   --postgres=postgres://<name>:
   <pwd>@localhost:5432/<db>?sslmode=disable
4. storj-sim network run

5 Discussion

We provide some discussion on possible solutions towards the DoS attack found in the Storj network, and the respective trade-offs to each potential solution. The Storj team was notified about the vulnerability we found in their dev./test network.

Vulnerability The DoS vulnerability stems from the inability of the server to close connections, which can be highly detrimental in the case of an operating satellite node. This can potentially be amended by making the server side (the satellite in our specific case) active instead of passive. Connections that have been idle for a certain period should be closed by the server independently of the client's behavior, thus preventing an uncontrolled increase of orphaned go routines.

dRPC library solution Storj establishes connections throughout its codebase via the dRPC library, which ultimately introduces many vulnerabilities throughout the system. This also means that solving the problem once within the library would solve it everywhere else.

Malicious client Closing connections that have been idle for a certain period would however not prevent the client to behave maliciously and opening as many connections as possible in a short amount of time. As a solution to this problem, we propose that the server filters out incoming connection requests based upon the corresponding timing information. The server will in essence only allow one connection within a certain time period. All other requests within the period are considered as malicious and would not be processed by the server. Nodes that exhibit this behaviour on a regular basis can even be excluded from the system by using some sort of blacklist.

6 Conclusions

The emergence of blockchain technologies and cryptocurrencies, has led to novel distributed storage systems which incentivize hosts to assure file availability. Distributed incentivization-based storage platforms, such as Sia and Storj, have become a potential alternative to cloud storage providers, enabling price competitiveness, and transparency on the storage of data.

In this paper, we have provided two contributions on one of the more prominent storage platforms Storj, namely: (i) an architectural overview, as well as (ii) a security exploration. The latter has resulted in the discovery of a DoS vulnerability within

Storj's dev./test environment, which we have experimentally evaluated to be highly feasible. The attack in question results in the inability of developers to access their test data and storage providers missing out on their payments. We have contacted Storj and their assessment is that Storj's production system (Multiple satellites running on load balanced clusters of servers.) is not vulnerable to such an attack.

Furthermore, we outline and discuss a multitude of approaches for solving the issue, mainly through the proper handling and closing of connections in satellite nodes.

In future work, we hope to conduct a more in-depth analysis on platforms such as Sia, Swarm, Filecoin (+ IPFS) and MaidSafe. Moreover, a systematic security analysis can facilitate a comparative evaluation of these systems. We hope this initiative will lead to the improvement of existing decentralized storage solutions, and steer future research.

References

[1] Lawrence Abrams. Amazon AWS outage shows data in the cloud is not always safe. https://www.bleepingcomputer.com/news/technology/amazon-aws-outage-shows-data-in-the-cloud-is-not-always-safe/, 2019.

[2] Nazanin Benisi, Mehdi Aminian, and Bahman Javadi. Blockchain-based decentralized storage networks: A survey. *Journal of Network and Computer Applications*, 162:102656, 04 2020.

[3] Nazanin Benisi, Mehdi Aminian, and Bahman Javadi. Blockchain-based decentralized storage networks: A survey. *Journal of Network and Computer Applications*, 162:102656, 04 2020.

[4] Arnar Birgisson, Joe Politz, Ulfar Erlingsson, Ankur Taly, Michael Vrable, and Mark Lentczner. Macaroons: Cookies with contextual caveats for decentralized authorization in the cloud. 01 2014.

[5] Fernando Bordignon and Gabriel Tolosa. Gnutella: Distributed system for information storage and searching model description. 08 2001.

[6] Carole Cadwalladr and Emma Graham-Harrison. Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach. https://www.theguardian.com/news/2018/mar/17/cambridge-analytica-facebook-influence-us-election, 2018.

[7] Huashan Chen, Marcus Pendleton, Laurent Njilla, and Shouhuai Xu. A survey on ethereum systems security: Vulnerabilities, attacks, and defenses. *ACM Computing Surveys*, 53:1–43, 06 2020.

[8] Sammy de Figueiredo. Storjattack, 2020. URL: https://github.com/Gitter2019-2020/StorjAttack.

[9] John Douceur. The sybil attack. pages 251–260, 01 2002.

[10] Elizabeth C Economy. The great firewall of China: Xi Jinping's internet shutdown. https://www.theguardian.com/news/2018/jun/29/the-great-firewall-of-china-xi-jinpings-internet-shutdown/, 2018.

[11] Egon Elbre. Storj test network, 2020. URL: https://github.com/storj/storj/wiki/Test-network.

[12] Ethereum. Ethereum is a global, open-source platform for decentralized applications., 2020. URL: https://ethereum.org/.

[13] Filecoin. Filecoin is a decentralized storage network designed to store humanity's most important information. https://filecoin.io/, 2020.

[14] Cloud Native Computing Foundation. A high-performance, open source universal rpc framework, 2020. URL: https://grpc.io/.

[15] Naveen Goud. Top 5 cloud security related data breaches! https://www.cybersecurity-insiders.com/top-5-cloud-security-related-data-breaches/, 2017.

[16] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. 08 2015.

[17] Janhvi Joshi and Rejo Mathew. *A Survey on Attacks of Bitcoin*, pages 953–959. 01 2020.

[18] Storj Labs. Storj: A decentralized cloud storage network framework (v3.0). https://storj.io/storj.pdf/, 2018.

[19] Storj Labs. Decentralized cloud storage is the future, 2020. URL: https://tardigrade.io/.

[20] Joseph Lewthwaite and Victoria Smith. Limewire examinations. *Digital Investigation*, 5, 09 2008.

[21] Jian Liang, Rakesh Kumar, and Keith Ross. Understanding kazaa. 06 2004.

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

[23] Lily Hay Newman. 1.2 billion records found exposed online in a single server. https://www.wired.com/story/billion-records-exposed-online/, 2019.

[24] Jorge Oliveira. Low orbit ion cannon, 2018. URL: https://github.com/NewEraCracker/LOIC.

[25] I. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8:300–304, 06 1960.

[26] Vincent Reniers, Dimitri Landuyt, Paolo Viviani, Bert Lagaisse, Riccardo Lombardi, and Wouter Joosen. Analysis of architectural variants for auditable blockchain-based private data sharing. pages 346–354, 04 2019.

[27] Virendra Soni. Top 6 secure cloud storage providers comparison in 2020. https://wire19.com/best-cloud-storage-providers-comparison/, 2020.

[28] Sia stats. Storage marketplace. https://siastats.info/storage_pricing/, 2020.

[29] Storj. drpc, 2020. URL: https://github.com/storj/drpc.

[30] Storj. Prerequisites, 2020. URL: https://storj.io/storj-operator-terms/.

[31] Ramesh Subramanian, Brian Goodman, Choon Ding, Sarana Nutanong, and Rajkumar Buyya. *Peer-to-Peer Networks for Content Sharing*, pages 28–65. 01 2005.

[32] Swarm. Storage and communication for a sovereign digital society. https://ethersphere.github.io/swarm-home/, 2020.

[33] RET2 SYSTEMS. A cryptocurrency heist, starring your web browser, 2019. URL: https://blog.ret2.io/2019/08/28/sia-coin-dns-rebinding/.

[34] Tardigrade.io. Decentralized object storage for developers. https://tardigrade.io/, 2020.

[35] Liam Tung. Google details 'catastrophic' cloud outage events: Promises to do better next time. https://www.zdnet.com/article/google-details-catastrophic-cloud-outage-events-promises-to-do-better-next-time/, 2019.

[36] Rahul Vokerla, Bharanidharan Shanmugam, Sami Azam, Asif Karim, Friso Boer, Mirjam Jonkman, and fahad faisal. An overview of blockchain applications and attacks. 11 2019.

[37] David Vorick and Luke Champine. Sia: Simple decentralized storage. https://sia.tech/sia.pdf/, 2014.

[38] Lloyd R. Welch and Elwyn R. Berlekamp. Error correction for algebraic block codes, 1986. US Patent US4633470A.

[39] Ren Zhang and Bart Preneel. Lay down the common metrics: Evaluating proof-of-work consensus protocols' security. pages 175–192, 05 2019.

[40] Xiaolu Zhang, Justin Grannis, Ibrahim Baggili, and Nicole Beebe. Frameup: An incriminatory attack on storj: A peer to peer blockchain enabled distributed storage system. *Digital Investigation*, 29, 03 2019.

[41] Shoshana Zuboff. *The Age of Surveillance Capitalism*. Publicaffairs, 2019. ISBN-13: 978-1-61039-569-4.

Publication 3

Off-chain State Channels in the Energy Domain

Publication Data

Clemens Brunner, Akash Madhusudan, Dominik Engel and Bart Preneel, "Off-chain state channels in the energy domain," 2021 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), Washington, DC, USA, 2021, pp. 1-5, doi: 10.1109/ISGT49243.2021.9372246.

Contributions

- I was the main author together with Clemens Brunner involved in Methodology, Writing - Original Draft, Resources, Visualization, and Conceptualization

Off-chain state channels in the energy domain

Clemens Brunner, Akash Madhusudan, Dominik Engel, and Bart Preneel

KU Leuven, ESAT/COSIC and imec, Leuven, Belgium and Center for Secure Energy Informatics, Salzburg University of Applied Sciences, Austria
{akash.madhusudan, svetla.nikova, bart.preneel}@esat.kuleuven.be
{clemens.brunner,dominik.engel}@en-trust.at

Abstract. Blockchain technology has attracted attention in the energy domain as a new decentralized infrastructure, with startups and researchers presenting new solutions to bring the benefits of decentralization to energy-related use cases. However, blockchains, or more specifically, public permissionless blockchains do not scale well. Scalability of blockchain solutions is often left as future work. Off-chain protocols such as payment channels, or in general, state channels are ways to improve the scalability of blockchain-based applications. The benefits of these state-of-the-art state channels in scaling blockchain-based applications in the energy domains have still not been investigated. Thus, we present a methodology to indicate if the various use cases in the energy domain benefit from state channels. Furthermore, we systematically assess these use cases by applying our proposed methodology. We found that state channels improve the scalability of various energy-related use cases, such as energy trading, and help in solving energy optimization problems.

1 Introduction

Blockchain technology was initially designed to enable monetary transactions (txs) by using digital currencies without the need of trusted third parties (TTPs) [19]. In addition to finance, blockchains can provide numerous benefits to other applications, and researchers and companies in the energy sector [15, 10, 4, 18] are working on new or enhanced protocols for various energy-related use cases, for instance electricity consumption optimization [10] or energy trading [16], to reap the benefits of decentralization. With the move of energy infrastructure towards smart grid (SG), bi-directional communication and electricity flows will allow a large number of renewable energy sources (RESs) to integrate. This increase in RESs will ultimately provide a market for local energy trading, or local energy markets. The shift from

designated suppliers to also local households acting as additional suppliers suggests decentralization, which presents a scope for integration of smart grids with blockchain technology. A problem with state-of-the-art public blockchain technology is that it does not scale [13].

The two most widely adopted public blockchains, i.e., Bitcoin and Ethereum, allow for about ten transactions (txs) [3] per second. In the energy domain, the scalability of blockchain-based applications is often left as future work, e.g. in [16], or a private instance of a blockchain is used.

Off-chain state channels are one method to increase the scalability of a blockchain-based application by reducing the number of necessary blockchain interactions [14]. In this paper we discuss the use of state channels for improving the scalability of blockchain-based applications in the energy domain. Our research question is as follows: *Which energy-related use cases benefit from off-chain state channels?* To answer this question, we provide the following contributions: Firstly, we present a methodology to analyze whether the scalability of a blockchain-based application improves by using state channels. Secondly, we apply our methodology to energy-related use cases and discuss if state channels can be applied to improve their scalability. Lastly, we present our results by listing all the energy-related use cases which benefit from state channels.

The paper is structured as follows: In Section 2 we provide a technical background followed by a presentation of our methodology in Section 3. In Section 4 we apply our methodology on blockchain-based use cases in the energy domain and present an evaluation. Finally, we summarize our work in Section 5.

2 Background and Related Work

This section provides the required technical background on blockchains and state channels, followed by an explanation of the differences in our approach compared to related work.

2.1 Blockchain and Scalability

A blockchain is an append-only, authenticated and distributed database. Signed txs are used to update the global state, which has to be agreed on by several nodes. The procedure to agree on a new state is called a consensus protocol [3]. Ethereum [24] introduced smart contracts, where txs also enable the deployment and execution of programmable code which changes the state of the system.

Smart contracts allow for generation of virtual tokens, which can be exchanged between participants. If tokens are attached to attributes, e.g., production date or used to represent a physical asset, they are considered to be non-fungible [20]. In contrast to fungible tokes, such as cryptocoins, it is not possible to perform arithmetic operations on non-fungible tokens. Fungible tokens require lower operational cost for storage and transfer. In this paper, we use the term *asset* for tokens or coins.

Finding consensus, without relying on a TTP, over a set of untrusted participants limits the scalability and hinders the wide spread adoption of blockchain technology. There are several ways to address this issue, e.g., changing the consensus mechanism, sharding or implementing a side chain [8]. Those improvements require either to create a new blockchain or to update existing ones.

In contrast to the adorementioned solutions, off-chain solutions can be created on top of an existing blockchain and do not change the trust assumptions of the underlying blockchain protocol. For a detailed explanation of off-chain solutions, we refer readers to [8].

2.2 State Channels

Off-chain state channels have three phases: (i) opening, (ii) off-chain modification and (iii) closing. To create a state channel, a portion of the blockchain state needs to be isolated and locked on-chain. This portion is called state deposit and is represented by assets. Deposits are created and locked by all participants $p_1, \ldots, p_n$ by sending an on-chain transaction.

To modify the state deposit, the agreement of all participants is necessary. In Figure 3.1 the modification of a state deposit SD_n is denoted with u_n. For this, signed messages are exchanged directly between the participants off-chain.

Each off-chain state is attached to a version number and signatures from all participants. Therefore, the creation of a new off-chain state depends on the responsiveness of the state channel participants. Hence, if a participant is offline or unresponsive, it is not possible to update the state and the state channel needs to be closed.

Closing a state channel requires the final state to be published on the blockchain. Usually, a predefined time delay gives the chance to all participants of a state channel to submit an off-chain state with a higher version number before the final state is accepted on the blockchain. This provides a level of integrity guarantee to the participants that the correct final state is used to close the channel and split the state deposit.

State channel can provide two additional properties to their underlying blockchain, but do not improve the private and anonymity of users unless specifically designed to do so [7].

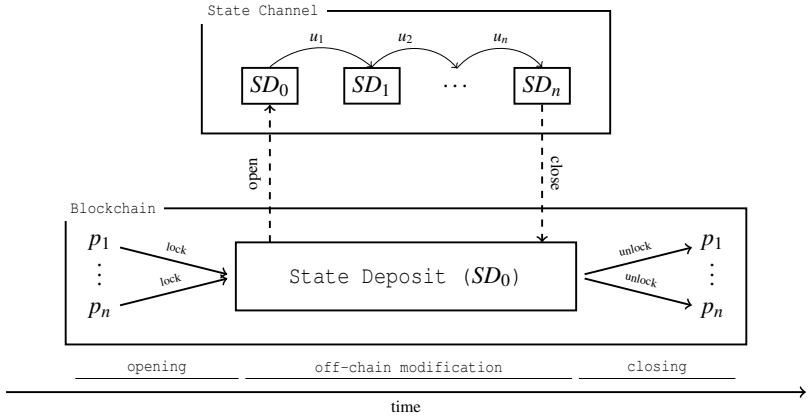


Figure 3.1: In this figure n participants create a state deposit by sending on-chain txs (lock). After this they can update the state deposit off-chain, which is illustrated with u_n. At the end they close the channel and unlock the deposit

Speed: Once a state channel is opened, agreement between a small set of participants can be achieved very quickly. The speed depends on the responsiveness of the participants and their network connection for exchanging off-chain messages.

Instant finality: For on-chain txs, users wait for confirmations before the tx is considered complete. In state channels, participants can submit the latest state, which is signed by all participants, to close and finalize the channel instantly. In case of disputes, the underlying blockchain provides integrity.

2.3 Related Work

To the best of our knowledge, no prior work performs an evaluation of state channels in the energy domain.

Related work about analyzing blockchain-based applications and providing a step-by-step guideline to decide whether a blockchain makes sense is described by Wüst and Gervais [25]. We will use this guideline to make the first evaluation for the use cases. In contrast to their work, we go one step further and check whether a state channel makes sense.

Evaluation and surveys of blockchain-based energy use cases are described in [2, 1]. We use those publications as a basis for selecting use cases. These works summarize the state-of-the-art of blockchain-based applications but do not deal with state channels.

3 Methodology

In this section we present the methodology to evaluate the applicability of state channels to blockchain-based applications.

For the evaluation we focus on a general question, i.e., *Does a state channel make sense?* We do not evaluate any extensions or combination of state channels [6]. We argue that if a state channel construction is not possible, the creation of a channel hub (or any other extension of state channels) would not provide additional benefits either.

3.1 Application Costs

In terms of cost evaluation, we express application cost as necessary on-chain txs, as that provides a clear estimation of the impact of state channels in increasing scalability. Other types of costs, such as the execution cost of tx or channel logic is outside the scope of our work.

To open a state channel with n participants, one transaction to create the smart contract, and n txs for all participants to lock and send a deposit to the state channel are necessary. Once a channel is opened the participants can exchange signed off-chain messages to agree on a state modification. If a participant wants to close a channel, in the best case only one transaction is necessary to submit the final state and unlock the deposit.

Hence, the minimum cost of a state channel for n participants is $n+2$ on-chain txs.

3.2 Methodology

Our methodology is sequential: we have five different steps, illustrated in Figure 3.2, that help us classify whether these use cases can benefit from state channels or not. If any of the steps in our methodology does not fit the use case, we stop evaluating it further.

Blockchain applied (BA): Here we evaluate for each use case if blockchain technology can be applied and what kinds of txs are needed. The discussion is based on the procedure presented in [25]. The main point that we focus on is whether blockchain is useful or not.

Asset exchange (AE) : After identifying the usefulness of blockchains, we look at all blockchain txs. Txs that change or transfer the ownership of assets are required to implement a state channel. If the txs are only needed to record or timestamp data, a state channel cannot provide any benefits.

Channel deposit (CD): A major requirement for state channels to work is a channel deposit, which can be used for instant and atomic payments and also serve as collateral to safeguard participants from malicious behavior. Hence, we only evaluate the use cases where the final ownership of assets needs to be modified by the participants.

Instant finality (IF): The use cases which require fast ownership changes, where it is not possible to wait until a transaction is confirmed (public blockchains incur a delay in the confirmation of txs because of their distributed nature), benefit significantly by using state channels.

On-chain tx reduction (RT): We discuss the number of necessary on-chain txs with and without a state channel. We argue that use cases where the number of on-chain txs can be reduced benefit by using state channels.

4 Evaluation of Use Cases in the Energy Domain

In this section we provide an overview of blockchain use cases in the energy domain and apply our methodology to evaluate whether or not state channels can be applied. The use case selection is based on the findings in [2, 1].

4.1 Energy Trading

Current energy trading is carried out in three different markets, namely, wholesale, retail and balancing market. As argued in [2], existing wholesale energy trading markets consist of slow procedures due to complexity added by requiring third-party intermediaries. Also, the small throughput of txs tend to be prohibitive to small-to-medium enterprises (SMEs). The EnerChain [17] framework uses a blockchain to enable energy trading on the wholesale market. Grid+ [5] aims to connect customers directly to electricity generators (GENs) and combine the wholesale market with the retail market.

BA: The blockchain acts as a decentralized marketplace, where GENs are allowed to create tokens and sell them, and energy suppliers or consumers are allowed to buy and use the produced electricity in the form of these tokens. This decentralization also allows such markets to avoid a TTP. Hence a blockchain is applicable in this use case.

AE: Electricity can be represented as a token. Please note that each token must be attached at least to a time-slot which identifies its production date. Hence, tokens from different time-slots or GENs are non-fungible. These tokens can be treated as assets, which require asset exchange to change ownership between participants.

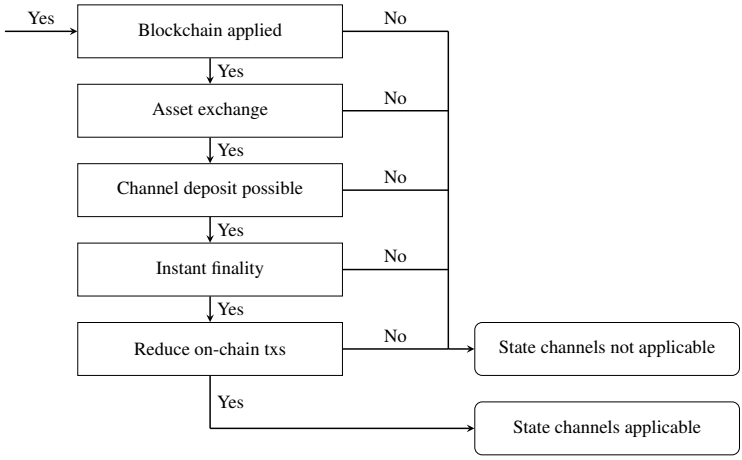


Figure 3.2: In this figure we depict how our procedure is applied to classify the applicability of state channels

CD: A channel deposit can be created by GENs and energy suppliers with electricity tokens. For energy trading, a channel deposit makes sense in existing markets. This is because the GENs and suppliers in such markets are well-known entities and frequently trade with each other.

IF: Due to the sheer volume of involved parties and limited time windows, this use case requires the instant finality provided by state channels to enable fast trades.

RT: In wholesale energy trading, where GENs sell electricity to multiple suppliers and SMEs and they trade the tokens again with each other, a state channel can reduce the load of txs on blockchain. Without state channels, all trades will be stored on-chain which adversely affects the transaction throughput due to the large amounts of txs waiting to be confirmed.

After applying our methodology, we conclude that state channels are applicable and beneficial for energy trading. In addition to providing instant finality for trade settlements, they also reduce the number of required on-chain txs. For instance, if m trades for n participants (GENs, suppliers or SMEs) are necessary to find the final distribution of electricity, $m * n$ on-chain transactions are needed. As mentioned in Section 3, with state-channels, only $n+2$ on-chain txs are required.

4.2 Local Energy Trading

Blockchain-based local energy trading is suggested by [15, 21, 16]. The idea of these energy trading platforms is to allow households, which have a RES, e.g., wind turbine, or photo-voltaic panels, to sell their surplus of produced electricity directly to a neighbor. This should reduce the electricity that is injected back to the main grid and increases the remuneration for the local producer.

BA: Since distributed households are allowed to trade electricity amongst each other in local energy trading, the use of blockchains directly compliments this use case.

AE: Similar to the previous use case, electricity can be represented as tokens, and asset exchange is required change ownership between participants.

CD: This use cases only satisfies our methodology when participants know each other beforehand, and if the energy tokens are traded more than once. Hence, in cases where these conditions are met, channel deposits are possible, i.e., between neighbours who frequently trade electricity amongst each other.

IF: Similar to wholesale trading, local energy trading can also benefit from instant transaction confirmation. Instant transaction confirmation give the prosumers confidence since they are instantly renumerated for the electricity they sell.

RT: The reduction of on-chain txs for energy trading also depends on the fact that the energy tokens are traded only once, e.g., from a prosumer to a consumer. In the local energy trading use case, where consumers only buy the tokens for their consumption but do not to sell it for a better price again, the setup of a state channel requires more txs than a single transfer. Therefore, we argue that a state channel does not reduce the on-chain costs/txs for this use case.

Since this use case does not satisfy the final step of our methodology, we conclude that state channels are not beneficial for local energy trading.

4.3 Energy Certificate Trading

A green energy certificate is an electronic document that represents the origin of produced electricity, the generation type and date, and is unified to 1 MWh, as described in the Directive 2009/28/EC [23]. These documents can be traded independently of the physical energy consumption. In [4], a private Ethereum blockchain and in [9], a public blockchain is suggested as the underlying technology for a trading platform.

BA: In this use case the blockchain is used to create, transfer the ownership and sell energy certificates. Those certificates can be traded between prosumers and

consumers [9] on the retail market and also between GENs and energy suppliers on the wholesale market [4].

AE: Certificate tokens are unique and classified as non-fungible tokens, because they depend on the production date and origin. Hence, asset exchange is required to change the ownership of these certificates.

CD: In this use case the channel deposit can be created with energy certificate tokens between the involved parties. Even though the deposit is not a requirement for this use case, we proceed with our methodology.

IF: In this use case, instant finality is not a requirement for the trade to function as typically the certificate is not traded multiple times. Hence, we stop evaluating this case further.

Since this use case does not satisfy the third step of our methodology, we conclude that state channels are not beneficial for energy certificate trading.

4.4 Electricity Consumption Optimization

In electricity consumption optimization or demand management, participants of a local energy community, e.g., households, submit strategies that optimize the supply and demand. Each participant is then allowed to commit a better solution based on the existing ones. An electricity consumption optimization can be applied within a smart grid where shift-able devices, e.g., heat pumps or electric vehicle charging stations, allow for shifting the consumption to a predefined time-slot. The goal is to be as independent as possible from the main grid. This is achieved by reducing the electricity (1) fed to the grid, and (2) withdrawn from the grid. In [10], a directed acyclic graph (DAG) and in [18], a blockchain is suggested to enable a community-driven optimization process.

BA: This use case does not require the use of blockchains, however as seen in [10, 18], the use case can benefit from the application of blockchains. The blockchain helps unknown participants to agree on a new optimized state while simultaneously removing the need of a TTP. Also, since this information is public, security issues such as non-repudiation can be avoided.

AE: Since the participant with optimal strategy needs to be incentivized, asset exchange (monetary transfer) is crucial in this use case if the incentivization involves cryptocurrency.

CD: If all the participants involved in the optimization of electricity usage know each other, channels can be established to make the incentivization process instantaneous.

IF: Since in electricity consumption optimization several strategies are submitted within a short time delay, state channels provide a solution for inherent confirmation delays of public blockchains, and hence benefit this use case.

RT: The participants submit several strategies that each require a separate transaction, and the frequency of submissions is high due to the incentive-driven nature of the system. By using state channels, the number of on-chain txs can significantly be reduced, hence we argue that for this use case a state channel makes sense.

We conclude that state channels are applicable and beneficial to this use case since it satisfies each step of our methodology.

4.5 Electric Vehicle Charging

Blockchain-based Electric Vehicle (EV) charging infrastructures are suggested by [12, 22] which generally require a car to purchase electricity from energy providers, e.g., prosumers, energy suppliers or charging station providers.

BA: In this use case the blockchain acts as a payment service between the car and the energy providers. The payment can be done by transferring electricity tokens or coins. The need of TTPs to facilitate these transactions is also removed.

AE: Electricity tokens or cryptocurrency are exchanged for the payment of consumed electricity. Electricity tokens, as described before, are non-fungible. If the tokens are only needed for the payment, they can be represented as fungible tokens such as cryptocurrency. Since payments involve the use of either of these types of tokens, asset exchange is required.

CD: If payments are done using cryptocurrency tokens, a channel deposit can be made with the EV charging station. As an example, channel deposit between the car and the energy providers can be seen as a voucher. The voucher is locked and the value is reduced with each charging process.

IF: EV charging typically benefits from instant finality provided by state channels because of the involved micropayments between the customer and EV charging stations. A latency in payment confirmation would lead to customer dissatisfaction as it would incur a volatile waiting time.

RT: A state channel construction can provide a reduction of on-chain transaction if the deposits allows for more than one charge. Micropayments are more efficient and cheap when done off-chain due to their low monetary value. In micropayments, the transaction fees would be a burden for the consumer as it might be equal to the actual payment itself.

After the application of our methodology to the use case of EV charging, we conclude that state channels are applicable and beneficial as it satisfies each step in our methodology.

4.6 Tariff Decisions

In this use case, electricity consumers can change their tariffs based on their energy consumption patterns. These pattern changes signify the tariff decisions of these customers. In [11], privacy-preserving tariff decision with blockchain-based smart contracts are presented since having a trusted entity mediating these decisions could lead to severe privacy threats for consumers, such as customer profiling.

BA: Blockchain can be applied in the use case of tariff decisions and used as a time stamping service. In addition to that, all decisions can also be logged on a blockchain for keeping a backlog of data and later using this data to optimize tariff decisions. Since this data is not stored in a central database, there is no single point of failure.

AE: For tariff decisions, there is no requirement of asset exchange. The blockchain is mainly used as an immutable untrusted source of information. Hence, the usage of state channels does not make any further contribution here and we stop investigating this use case further.

This use case does not satisfy the second step of our methodology, we conclude that state channels are not beneficial.

5 Conclusion

In this paper, we motivated the need for off-chain solutions in the energy domain. We described energy use cases presented by peer-reviewed publications, where blockchains are proposed as an underlying trust layer. We presented a methodology and sequentially applied it to the use cases to evaluate whether state channels are applicable to them and why. We found out that state channels can be applied to energy trading and for electricity consumption optimization. This work provides a solid baseline for evaluating the benefits of state channels and blockchain in the energy domain. It paves a path for further evaluation of state channels in the energy sector. As a next step, an implementation and evaluation of operating costs of a state channel in the energy domain is proposed.

Table 3.1: Comparison of blockchain-based energy use cases according to defined criteria.

Use case	BA	EA	CD	IF	RT
Energy Trading [17, 5]	✓	✓	✓	✓	✓
Local Energy Trading [16, 15, 21]	✓	✓	✓	✓	✗
Energy Certificate Trading [4, 9]	✓	✓	✓	✗	✓
Electricity Consumption Optimization [10, 18]	✓	✓	✓	✓	✓
EV Charging [12, 22]	✓	✓	✓	✓	✓
Tariff Decisions [11]	✓	✗	✗	✗	✗

References

[1] Simon Albrecht, Stefan Reichert, Jan Schmid, Jens Strüker, Dirk Neumann, and Gilbert Fridgen. Dynamics of Blockchain Implementation - A Case Study from the Energy Sector. In *51st Hawaii International Conference on System Sciences*, Hawaii, USA, 2018. Curran Associates, Inc.

[2] Merlinda Andoni, Valentin Robu, David Flynn, Simone Abram, Dale Geach, David Jenkins, Peter McCallum, and Andrew Peacock. Blockchain technology in the energy sector: A systematic review of challenges and opportunities. *Renewable and Sustainable Energy Reviews*, 100:143–174, feb 2019.

[3] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. Consensus in the Age of Blockchains. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 183–198, New York, NY, USA, 2019. ACM.

[4] J. Alejandro F. Castellanos, Debora Coll-Mayor, and José Antonio Notholt. Cryptocurrency as Guarantees of Origin: Simulating a Green Certificate Market with the Ethereum Blockchain. In *5th IEEE International Conference on Smart Energy Grid Engineering*, pages 367–372, Oshawa, Canada, 2017. IEEE.

[5] Consensys. Grid+: Welcome to the future of energy. https://gridplus.io/whitepaper, 2017. accessed on April 2020.

[6] Stefan Dziembowski, Lisa Eckey, Sebastian Faust, and Daniel Malinowski. Perun: Virtual Payment Hubs over Cryptocurrencies. *Proceedings - IEEE Symposium on Security and Privacy*, 2019-May:106–123, 2019.

[7] Matthew Green and Ian Miers. Bolt: Anonymous payment channels for decentralized currencies. In *2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 473–489, 2017.

[8] Lewis Gudgeon, Pedro Moreno-sanchez, Stefanie Roos, Patrick Mccorry, and Arthur Gervais. SoK: Layer-Two Blockchain Protocols. In *Financial Cryptography and Data Security*, Kota Kinabalu, Sabah, Malaysia, 2020. Springer.

[9] Fabian Knirsch, Clemens Brunner, Andreas Unterweger, and Dominik Engel. Decentralized and Permission-less Green Energy Certificates with GECKO. *Energy Informatics*, 3(2):1–17, 2020.

[10] Fabian Knirsch, Oliver Langthaler, and Dominik Engel. Trust-less Electricity Consumption Optimization in Local Energy Communities. *Energy Informatics*, 2(1):1–12, 2019.

[11] Fabian Knirsch, Andreas Unterweger, Günther Eibl, and Dominik Engel. Privacy-Preserving Smart Grid Tariff Decisions with Blockchain-Based Smart Contracts. In Wilson Rivera, editor, *Sustainable Cloud and Energy Services: Principles and Practices*, chapter 4, pages 85–116. Springer International Publishing, Cham, Switzerland, 2017.

[12] Fabian Knirsch, Andreas Unterweger, and Dominik Engel. Privacy-preserving Blockchain-based Electric Vehicle Charging with Dynamic Tariff Decisions. *Journal on Computer Science - Research and Development (CSRD)*, 33(1):71–79, 2018.

[13] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 17–30, 2016.

[14] Patrick McCorry, Chris Buckland, Surya Bakshi, Karl Wüst, and Andrew Miller. You sank my battleship! A case study to evaluate state channels as a scaling solution for cryptocurrencies, 2018. accessed on March 2020.

[15] Esther Mengelkamp, Johannes Gärttner, Kerstin Rock, Scott Kessler, Lawrence Orsini, and Christof Weinhardt. Designing microgrid energy markets: A case study: The Brooklyn Microgrid. *Applied Energy*, 210:870–880, 2018.

[16] Esther Mengelkamp, Benedikt Notheisen, Carolin Beer, David Dauer, and Christof Weinhardt. A blockchain-based smart grid: towards sustainable local energy markets. *Computer Science - Research and Development*, 33(1):207–214, 2018.

[17] Michael Merz. European energy trading firms test peer-to-peer trading over the blockchain. https://enerchain.ponton.de/index.php/21-enerchain-p2p-trading-project/, 2017. accessed on April 2020.

[18] Eric Munsing, Jonathan Mather, and Scott Moura. Blockchains for decentralized optimization of energy resources in microgrid networks. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pages 2164–2171, Mauna Lani, HI, USA, 2017. IEEE.

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

[20] Ferdinand Regner, Nils Urbach, and André Schweizer. NFTs in Practice – Non-Fungible Tokens as Core Component of a Blockchain-based Event Ticketing Application. *40th International Conference on Information Systems (ICIS 2019)*, pages 1–17, 2019.

[21] Pierluigi Siano, Giuseppe De Marco, Alejandro Rolan, and Vincenzo Loia. A Survey and Evaluation of the Potentials of Distributed Ledger Technology for Peer-to-Peer Transactive Energy Exchanges in Local Energy Markets. *IEEE Systems Journal*, 13(3):3454–3466, 2019.

[22] Zhou Su, Yuntao Wang, Qichao Xu, Minrui Fei, Yu Chu Tian, and Ning Zhang. A Secure Charging Scheme for Electric Vehicles with Smart Communities in Energy Blockchain. *IEEE Internet of Things Journal*, 2018.

[23] The European Parliament and the Council of the European Union. Directive 2009/28/EC of the European Parliament and of the Council of 23 April 2009 on the promotion of the use of energy from renewable sources, 2009.

[24] Gavin Wood. Ethereum: A Secure Decentralised Generalised Transaction Ledger. Technical report, Ethereum, 2017.

[25] Karl Wüst and Arthur Gervais. Do you need a blockchain? https://eprint.iacr.org/2017/375/, 2017. Cryptology ePrint Archive, Paper 2017/375.

Publication 4

Billing Models for Peer-to-Peer Electricity Trading Markets with Imperfect Bid-Offer Fulfillment

Publication Data

Akash Madhusudan, Fairouz Zobiri and Mustafa A. Mustafa, "Billing Models for Peer-to-Peer Electricity Trading Markets with Imperfect Bid-Offer Fulfillment," 2022 IEEE International Smart Cities Conference (ISC2), Pafos, Cyprus, 2022, pp. 1-7, doi: 10.1109/ISC255366.2022.9922530.

Contributions

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

Billing Models for Peer-to-Peer Electricity Trading Markets with Imperfect Bid-Offer Fulfillment

Akash Madhusudan, Fairouz Zobiri, and Mustafa A. Mustafa

KU Leuven, ESAT/COSIC, Belgium, KU Leuven, ESAT/ELECTA, Belgium, and University of Manchester
{akash.madhusudan}@esat.kuleuven.be

Abstract. This paper proposes four new billing models for peer-to-peer electricity trading markets that take into account electricity volume deviations of market participants from their bids/offers. These billing models incorporate different cost sharing mechanisms so that (i) the costs incurred due to these deviations are minimal for consumers and prosumers and (ii) include non peer-to-peer participants as well. The former is achieved by designing cost sharing mechanisms which split the cost socially, while the latter is achieved by introducing a mid-market that clears all the available supply from the P2P market. Through simulations of a small-scale community, we have demonstrated the effectiveness of our billing models in significantly increasing prosumers' rewards and reducing consumers' bills.

1 Introduction

Smart grid is an electrical grid enhanced with two-way electricity and communication flow capabilities [6]. It is equipped with various smart devices which can sense and report measurements from all parts of the grid such that informed decisions can be made timely. Among these devices, smart meters are thought to be the ones that will help transform user behaviour. They can measure fine-grained electricity consumption and production volumes of users and make such data available for new smart grid applications [15].

Peer-to-peer (P2P) electricity trading is one such application that has gained attention in recent years due to its potential to engage users in grid balancing [14]. It allows electricity to be traded between users who can determine the trading price based on their current demand/supply needs. The trading price is usually determined at an auction to which users submit their bids/offers in advance [3]. When there is more demand for electricity, the trading price is high, so that more prosumers are encouraged to sell

electricity, while consumers are encouraged to consume less electricity. Similarly, if there is more supply available, the trading price is low encouraging prosumers to keep their excess electricity, while consumers to increase their demand to take advantage of the low price [8].

Most P2P markets require users to submit bids/offers for future trading periods in advance (ranging from a day to 30 minutes in advance). To do so, users will have to accurately predict how much demand or supply they will need or provide. However, these predictions might not always be accurate or users might require/have unexpected extra demand/supply. As a result, users might not be able to accurately fulfil their demand/supply commitments at the P2P market; hence, their actual demand/supply volumes during the trading slot (measured by their smart meters) might deviate from their committed demand/supply volumes at the P2P market (i.e., the volumes submitted with their bids/offers and accepted at the P2P market). These deviations can disrupt the operation of the grid and increase the grid balancing cost [5], which could be passed on to the users. Therefore, to minimise these costs, there is a need for billing models that incentivise prosumers/consumers to minimise their deviations. However, existing work on P2P markets have largely ignored this issue. They have either assumed a perfect fulfilment of the committed volumes [2], did not consider billing models that incorporate these deviations [13] or proposed symmetric imbalance charge mechanisms which penalise market participants irrespective of the direction of their energy imbalance deviation [4]. A simple billing mechanism was proposed in [9], however, the focus of the work is on calculating bills privately. Ausubel's clinching auction has been modified in [10] to guarantee truthful user participation. The proposal, however, does not include any billing models.

To address this gap, we propose four novel billing models that incorporate these deviations and split the cost associated with these deviations amongst users fairly; without any of them being heavily penalised. Specifically, the novel contributions of this paper are:

- We design four novel billing models that take into account the individual deviations of P2P market participants and split the cost incurred amongst the P2P participants. Each billing model has a distinct cost splitting mechanism: (i) individual cost split, where individual users are responsible for their own individual deviations; (ii) social cost split, where individual deviations of prosumers/consumers are aggregated and the total supply/demand cost is split amongst deviating prosumers/consumers socially, (iii) universal cost split, where the total supply and demand deviations are used to derive the total deviation per trading slot. Based on this total deviation, the deviating P2P participants are penalized. (iv) universal cost split with mid-market, where non-P2P participants compensate for the deviations of P2P consumers/prosumers; hence, reducing the penalty for the deviating participants. Additionally, by utilizing the mid-market,

non-P2P participants also trade amongst each other at a better price than the retail market.

- We run simulations on a small-scale community (15 participants: 10 consumers and 5 prosumers) to demonstrate the effectiveness of our proposed billing models in reducing the cost for consumers and increasing the profits of prosumers.

The rest of the paper is organised as follows; in Sec. 2 we depict our novel billing models and describe them in detail. In Sec. 3 we evaluate our billing models and show their effect on consumer bills, prosumer rewards and supplier income. Finally, in Sec 4 we conclude our work and list some interesting future work venues.

2 Billing Models

In this section, we first provide a billing model used in a typical retail market, before proposing four new billing models dedicated to a P2P market that deal with the deviations of individual prosumers/consumers. By deviations we mean the difference between the volumes that these participants commit at the P2P market and the volumes they actually fulfil. Three of these billing models use only the Retail Market (RM) as a back-up option to deal with the deviations, while the last billing model uses an additional market, called a mid-market, as a default back-up market for dealing with deviations before falling back, if necessary, to the RM. These billing models use three different methods for splitting the cost incurred due to the individual deviations of prosumers/consumers. These methods are called individual, social and universal cost split. These billing models are detailed next. Notations are listed in Table 4.1.

2.1 Billing Model for Retail Markets – the status quo

Most liberalised electricity retail markets allow prosumers and consumers to trade only with suppliers they have contracts with. The tariff that suppliers use to buy any excess electricity supplied to the grid by the prosumers is regulated and set to a fixed price by the market regulators of each country. This tariff is called Feed-in-Tariff (FiT). No electricity trading between consumers and prosumers is allowed. There are no or little incentives for consumers/prosumers to change their load profiles. A summary of the billing model used in a typical retail market is given below as well as shown in Alg. 5.

- Consumers buy electricity only from their suppliers at a retail buy price. Retail buy prices are determined by the suppliers; suppliers usually have several tariffs consumers can choose from; these tariffs are set such that they are competitive compared to the tariffs offered by other suppliers.

Table 4.1: Abbreviations and Notations

Symbol	Description
P2P, MM	Peer-to-Peer, Mid-Market
TD	Total deviation
InDev_x	Individual deviation of a participant
InDev_i	Individual deviation of a consumer
InDev_j	Individual deviation of a prosumer
TSD	Total supply deviation
TDD	Total demand deviation
TP, RP	Trading price, Retail price
MMP_p, MMP_c	Mid-market sell price, Mid-market buy price
FiT	Feed-in tariff
C_n	Total no. of Non-P2P consumers
C_i	Individual consumer
$P2P_n^c$	Total no. of P2P consumers
$P2P_i^c$	Individual P2P consumer
C_{dem}	Individual demand of Non-P2P consumer
T_{dem}^c	Total demand of Non-P2P consumers
C_{dem}^{P2P}	Individual demand of P2P consumer
T_{dem}^{P2P}	Total demand of P2P consumers
P_n	Total no. of Non-P2P prosumers
T_{over}^c	Total volume over-consumed
$P2P_{over}^c$	Total no. of P2P prosumers over-consuming
T_{under}^c	Total volume under-consumed
$P2P_{under}^c$	Total no. of P2P prosumers under-consuming
P_j	Individual prosumer
$P2P_n^p$	Total no. of P2P prosumers
$P2P_j^p$	Individual P2P prosumer
P_{sup}	Individual supply of Non-P2P prosumer
T_{sup}^p	Total supply of Non-P2P prosumers
P_{sup}^{P2P}	Individual supply of P2P prosumer
T_{sup}^{P2P}	Total supply of P2P prosumers
T_{over}^p	Total volume over-supplied
$P2P_{over}^p$	Total no. of P2P prosumers over-supplying
T_{under}^p	Total volume under-supplied
$P2P_{under}^p$	Total no. of P2P prosumers under-supplying
S_n	Total suppliers
S_k	Individual supplier
S_k^{inc}	Individual supplier income
S_k^{exp}	Individual supplier expenditure
S_k^{bal}	Individual supplier balance

- Prosumers sell electricity only to suppliers at a fixed price, the FiT. In some countries, FiT is set to a different value depending on the type and size of the electricity source used to generate the electricity at the prosumer side.

Algorithm 5 Billing Model for Retail Markets [7]

Input: Consumption (C_{dem}), production (P_{sup}), RP, FiT
Output: C bill, P reward, Supplier (S) balance

1: C bill = $C_{dem} \times$ RP
2: P reward = $P_{sup} \times$ FiT
3: S balance = C bill - P reward

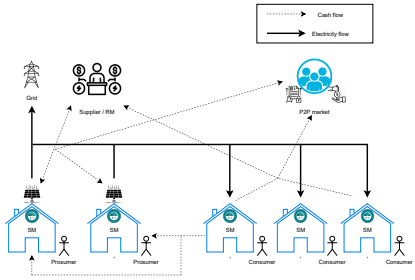


Figure 4.1: P2P market with RM as a fall-back option

- Suppliers sell electricity to their customers (contracted consumers) and buy back any electricity their customers (contracted prosumers) feed back to the grid.

2.2 Billing Models for P2PM with RM as Back-up

P2P trading markets allow prosumers and consumers to trade electricity among each other. They involve the following steps.

- Consumers/Prosumers submit bids/offers for electricity;
- A market operator (e.g., trading platform) clears the P2P market. It determines the total cleared volume of electricity at the market, the market clearance price and the consumers/prosumers whose bids/offers have been accepted. Consumers/Prosumers whose bids have not been accepted use the RM to meet their needs as the RM acts as a fall-back option.
- At the end of the trading period, data from the smart meters of consumers/prosumers is used to calculate their bills/rewards.
- If the data from the smart meters of prosumers/consumers whose bids have been accepted is not the same as their respective committed value in the bid/offer, the difference (deviation) is compensated by the RM. This compensation could come with an extra cost for some of the prosumers/consumers.

The cost associated with compensating the deviations of the individual prosumers/-consumers will have to be distributed amongst all (some) of the prosumers/consumers. There could be different mechanisms to share these costs amongst prosumers/consumers. Next, we propose three different methods of how these costs could be shared: individual, social and universal cost split. An example P2P market with a RM as a fall-back option is shown in Fig. 4.1.

Billing with individual cost split

In this billing model, individual consumers/prosumers who partake in the P2P market bear the cost of their respective deviations from their committed bids/offers. The positive deviations are traded at the RM while the negative deviations are compensated from the RM. This billing model is depicted in our paper [7] and described below.

- Positive deviations of prosumers (i.e., a prosumer supplies more volume than they committed) are sold for the FiT to their respective supplier. Positive deviations of consumers (i.e., a consumer consumes more volume than they committed) are bought for the RP from their respective supplier.
- Negative deviations of prosumers (i.e., a prosumer supplies less volume than they committed) are compensated by the RM – prosumers buy their individual negative deviation at the RM for the RP and sell it at the P2P market for the TP in order to fulfill their commitment.
- Negative deviations of consumers are also compensated by the RM – consumers buy all their individual committed value at the P2P market at the TP, but since they have not managed to consume all of their bought value, they sell the deviation immediately to their respective supplier at the FiT price.
- Suppliers sell electricity at RP to consumers who consumed more volume than they bought, and to prosumers who supplied less volume than they committed at the P2P market.
- Suppliers buy electricity at FiT from prosumers who supply more volume than they committed, and from consumers who could not consume all the volume they bought at P2P market.
- Consumers/prosumers whose bids were not accepted at the P2P market trade with their suppliers at the RP and FiT.

Billing with social cost split

In this billing model, the individual deviations of prosumers are aggregated and the cost of the total supply deviation is socially split among the prosumers. Similarly, the individual deviations of consumers are aggregated and the cost of the total demand deviation is socially split among the consumers. This billing model is depicted in our paper [7] and described below.

- All individual deviations of the prosumers are aggregated to compute the total supply deviation.
- If the total supply deviation is equal to zero, then there is no collective supply deviation, hence all prosumers are paid at the P2P price for all their supplied volumes regardless of their individual deviations. Prosumers who oversupplied benefit as they sell even their individual deviation at the P2P price rather than the FiT. Prosumers who undersupplied benefit too as they do not have to compensate their individual deviation from RM.
- If the total supply deviation is positive (negative), then the total deviation of prosumers who under-supplied (over-supplied) partially compensate for the prosumers who over-supplied (under-supplied); Prosumers who under-supplied sell all their supplied electricity at the P2P price regardless of their individual deviation, while the prosumers who over-supplied reduce their individual deviation with a share of the total deviation of prosumers who under-supplied. Thus, prosumers either sell all their individual deviations at P2P price or reduce their deviations which they need to compensate for from the RM.
- Similarly, if the total demand deviation is positive (negative), then the total deviation of consumers who under-consumed (over-consumed) partially compensate for the consumers who over-consumed (under-consumed); Consumers who under-consumed buy all their electricity at the P2P price regardless of their individual deviation, while the consumers who over-consumed reduce their individual deviation with a share of the total deviation of consumers who under-consumed.
- Suppliers sell/buy electricity only to/from those prosumers/consumers whose individual deviation is in the same direction as the total supply/demand deviation.

Billing model with universal cost split

This billing model aggregates all the individual deviations of all P2P market participants (prosumers and consumers) and the cost of the total deviation is socially split amongst

the prosumers (consumers) whose individual deviation is in the same direction as the total deviation. This billing model is depicted in our paper [7] and described below.

- All individual deviations are aggregated to calculate the total deviation of the P2P market.
- If the total deviation is equal to zero, all prosumers (consumers) sell (buy) at the P2P trading price, regardless of their individual deviations.
- If the total deviation is positive (total supply is more than the total demand), then all consumers buy all their consumed volume at the trading price, regardless of their individual deviation, whereas prosumers proceed as in the social split case, splitting the reward calculated based on the total deviation.
- If the total deviation is negative (total supply is less than the total demand), then all prosumers sell all their supplied volume at the P2P trading price, regardless of their individual deviation, whereas the consumers proceed as in the social split case, splitting the cost calculated based on the total deviation.
- Suppliers only sell/buy volumes equal to the total deviation of the P2P market.

2.3 Billing Models for P2PM with Mid-Market as Back-up

In the previous billing models, the RM was the first and only back-up option for prosumers/consumers when they had to trade or compensate for their deviations. The prosumers/consumers whose bids were not accepted at the P2P market also trade only at the RM. To maximise the volumes traded between prosumers and consumers there should be a follow-up market for the unsuccessful bids at the P2P market. In this paper, we assume that there is a Mid-Market (MM) where volumes of users with unsuccessful bids are matched. We assume that the electricity at this market is traded at mid-market prices. These mid-market prices are calculated based on the trading, retail and FiT prices; similarly to the market models proposed in [11]. The prices are set such that prosumers and consumers are always better off trading at the P2P market. However, if they are not successful at the P2P market, then they automatically trade at the MM. Only volumes that are still uncleared after the MM are traded at the RM. The inclusion of such MM models increases the local supply-demand match provided by the prosumers and cosnumers and minimises volumes traded at the RM. This billing model is depicted in our paper [7].

3 Evaluation

We evaluate our billing models by running simulations on a small-scale use-case to calculate profits and bills for prosumers and consumers, respectively and the balances of the suppliers.

3.1 Use-case

Our use-case consists of a small community of 15 market participants (5 prosumers and 10 consumers) contracted with three suppliers: Supplier A, Supplier B and Supplier C. The customers of these suppliers are allocated as follows: Supplier A – 3 customers (2 consumers and 1 prosumer), Supplier B – 5 customers (3 consumers and 2 prosumers) and Supplier C – 7 customers (5 consumers and 2 prosumers). Each trading slot has a duration of one hour.

In scenarios with the RM only, prosumers/consumers do not submit any bids. Their rewards and bills are calculated based on the volumes of electricity they supply/consume over each trading slot using the retail prices they are contracted with and FiT. Suppliers buy all of the supply of their contracted prosumers as well as sell to their contracted consumers all of their demand volumes. In scenarios with a P2P trading market, a double auction mechanism is used to clear the P2P market. It determines a unified P2P trading price, the cleared supply and demand volumes and the auction winners. If MM is not present, the RM plays the role as the only fall-back option for prosumers and consumers. If MM is present, then the MM is the first fall-back option for the P2P market participants. The MM sell and buy price is determined by a function of the P2P price, average RM price and the FiT.

3.2 Data Generation

We randomly generate our data by utilizing the findings of Abidin et al. [1]. In their paper, they used a real dataset where the total generation of solar electricity was 2382 MW on 5/5/2016 between 13:00h and 13:30h. A Belgian household, on average, consumed 0.637 kW of electricity during that time slot [12]. Additionally, we assume that each prosumer has a solar panel installed at their home, with a capacity of 6 kW. Based on the chosen efficiency factor of the solar panels, i.e., 90%, we calculate the amount of excess electricity each prosumer has per trading slot randomly. Based on the aforementioned assumptions, we deduce that each consumer places a bid to consume electricity in the range of 0.4 – 1.1 kWh and each consumer places an offer to supply electricity in the range of 0.6 – 1 kWh per trading slot. From these ranges, we randomly assign a predicted demand/supply value to each

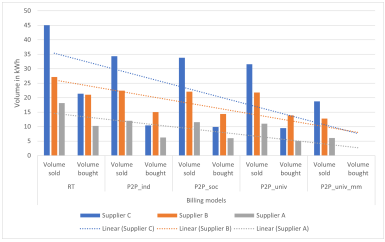


Figure 4.2: Suppliers' trading volumes at RT and P2P billing models

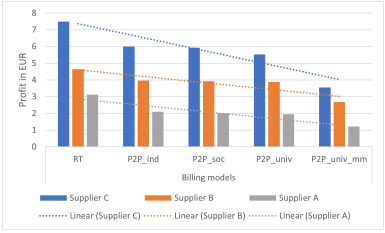


Figure 4.3: Suppliers' profits at RT and P2P billing models

consumer/prosumer. Once the predicted values are assigned, we assign fixed deviations that each consumer/prosumer would have as follows. The consumers/prosumers have a zero deviation from their predicted values in 10% of the trading slots. The consumers/prosumers have a high ($>$ +/- 0.1 kWh) deviation from their predicted values in 20% of the slots. The consumers/prosumers have a medium deviation (between +/- 0.05 to 0.1 kWh) from their predicted values in 20% of the trading slots. The consumers/prosumers have a low deviation ($\leq$ +/- 0.05 kWh) from their predicted values in 50% of the slots. Using these deviations, we deduce the volume the consumers/prosumers actually consumed/delivered. In our data, the positive and negative deviations are distributed equally.

We set the retail electricity price of suppliers to 0.20€/kWh, 0.21€/kWh and 0.19€/kWh for Supplier A, B and C, respectively. The FiT is set to 0.05€/kWh for each supplier. Additionally, based on the double auction algorithm provided in [1], we determine the auction winners in each trading slot along with the trading price for the respective slot. Finally, from this trading price we determine the mid-market buying/selling price by taking the average of the P2P trading price and the average retail/FiT price.

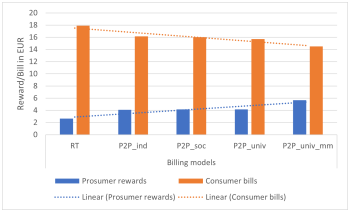


Figure 4.4: Difference in Bills/Rewards of all participants

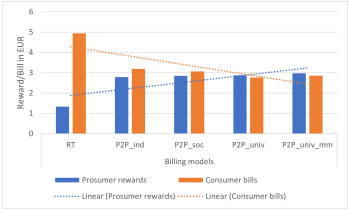


Figure 4.5: Difference in Bills/Rewards of P2P participants

3.3 Simulation Results

We simulate the demand/supply of each of the market participants for 12 trading slots (corresponding to half a day) and calculate the rewards and bills of prosumers and consumers along with suppliers' profits based on our billing models.

As can be seen in Fig. 4.2, having a P2P market reduces the volume of electricity traded at the RM significantly. For instance, Supplier C loses $\sim 25\%$ of the electricity volume sold while transitioning from RM to P2P. This drop is more severe when comparing RM to P2P markets along with mid-market, where Supplier C loses $\sim$ 60% of electricity volume sold. On the other hand, the volume of electricity bought by the suppliers' also reduces when transitioning from RM to P2P market, eventually falling to 0 when mid-market is introduced. As volumes traded by the suppliers' drop, we observe that the suppliers' income also reduces. This can be seen in Fig. 4.3.

Fig. 4.4 depicts the benefits for prosumers/consumers when transitioning from RM to P2P markets. Both, the rewards of prosumers and bills of consumers rise and fall respectively when moving from RM to P2P. However, between each P2P billing model, not much difference is noticed except when MM is introduced. For instance, the average reward of a prosumer doubles when we compare RM to P2P universal split with MM. In comparison, such a significant change is not observed in the bills of consumers.

Fig. 4.5 depicts that P2P billing models have a significant effect on prosumer/consumer

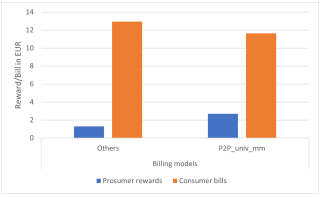


Figure 4.6: Difference in Bills/Rewards of non-P2P participants

rewards/bills; however, P2P billing models when compared to each other do not introduce much change. Fig. 4.6 shows how the introduction of MM in our billing models benefits everyone, including and mainly the prosumers/consumers with bids not accepted at the P2P market. Since the prosumers/consumers who do not get selected to trade on the P2P market have to trade on the RM, their profits do not change unless MMs are introduced.

4 Conclusions and Future Work

In this paper, we devise novel P2P billing models to incentivise P2P participants. Our evaluations show how these proposed billing models could have a significant impact on the volumes traded at the P2P market and RM; thus on the bills of consumers, rewards of prosumers and profits of suppliers. We also show how the introduction of MM significantly increases the profits of the prosumers who do not get selected to trade on the P2P market. Additionally, prosumers/consumers who trade on the P2P market also gain significant advantages with the introduction of MM.

As a future work, we plan to run large-scale simulations on a larger number of market participants and longer duration to see the full effect of these models on the market participants. We also plan to improve the models so that they are less privacy invasive.

References

[1] Aysajan Abidin, Abdelrahaman Aly, Sara Cleemput, and Mustafa A Mustafa. An MPC-based privacy-preserving protocol for a local electricity trading market. In *Int. Conf. on Cryptology and Network Security*, pages 615–625, 2016.

[2] Aysajan Abidin, Abdelrahaman Aly, Sara Cleemput, and Mustafa A Mustafa. Secure and privacy-friendly local electricity trading and billing in smart grid. *arXiv preprint arXiv:1801.08354*, 2018.

[3] Timothy Capper, Anna Gorbatcheva, Mustafa A Mustafa, Mohamed Bahloul, Jan Marc Schwidtal, Ruzanna Chitchyan, Merlinda Andoni, Valentin Robu, Mehdi Montakhabi, Ian J Scott, et al. Peer-to-peer, community self-consumption, and transactive energy: A systematic literature review of local energy market models. *Renewable and Sustainable Energy Reviews*, 162:112403, 2022.

[4] Timothy Capper, Jaise Kuriakose, and Maria Sharmina. Impact of energy imbalance on financial rewards in peer-to-peer electricity markets. *IEEE Access*, 10:55235–55254, 2022.

[5] Viktorija Dudjak, Diana Neves, Tarek Alskaif, Shafi Khadem, Alejandro Pena-Bello, Pietro Saggese, Benjamin Bowler, Merlinda Andoni, Marina Bertolini, Yue Zhou, et al. Impact of local energy markets integration in power systems layer: A comprehensive review. *Applied Energy*, 301:117434, 2021.

[6] Xi Fang, Satyajayant Misra, Guoliang Xue, and Dejun Yang. Smart grid — the new and improved power grid: A survey. *IEEE Communications Surveys & Tutorials*, 14(4):944–980, 2012.

[7] Akash Madhusudan, Fairouz Zobiri, and Mustafa A. Mustafa. Billing models for peer-to-peer electricity trading markets with imperfect bid-offer fulfillment. In *2022 IEEE International Smart Cities Conference (ISC2)*, pages 1–7, 2022.

[8] Jan Marc Schwidtal, Proadpran Piccini, Matteo Troncia, Ruzanna Chitchyan, Mehdi Montakhabi, Christina Francis, Anna Gorbatcheva, Timothy Capper, Mustafa A Mustafa, Merlinda Andoni, et al. Emerging business models in local energy markets: A systematic review of peer-to-peer, community self-consumption, and transactive energy models. *SSRN*, 2022.

[9] Robbie Thandi and Mustafa A. Mustafa. Privacy-enhancing settlements protocol in peer-to-peer energy trading markets. IEEE, 2 2021.

[10] Georgios Tsaousoglou, Konstantinos Steriotis, Nikolaos Efthymiopoulos, Prodromos Makris, and Emmanouel Varvarigos. Truthful, practical and privacy-aware demand response in the smart grid via a distributed and optimal mechanism. *IEEE Transactions on Smart Grid*, 11(4):3119–3130, 2020.

[11] Wayes Tushar, Tapan Kumar Saha, Chau Yuen, Thomas Morstyn, H Vincent Poor, Richard Bean, et al. Grid influenced peer-to-peer energy trading. *IEEE Transactions on Smart Grid*, 11(2):1407–1418, 2019.

[12] VREG. Aansluitingen, meters en data. https://www.vreg.be/nl/aansluitingen-meters-en-data, 2016.

[13] Yunpeng Wang, Walid Saad, Zhu Han, H Vincent Poor, and Tamer Başar. A game-theoretic approach to energy trading in the smart grid. *IEEE Transactions on Smart Grid*, 5(3):1439–1450, 2014.

[14] Chenghua Zhang, Jianzhong Wu, Yue Zhou, Meng Cheng, and Chao Long. Peer-to-peer energy trading in a microgrid. *Applied Energy*, 220:1–12, 2018.

[15] Jixuan Zheng, David Wenzhong Gao, and Li Lin. Smart meters in smart grid: An overview. In *2013 IEEE Green Technologies Conference (GreenTech)*, pages 57–64, 2013.

Publication 5

Fully Privacy-Preserving Billing Models for Peer-to-Peer Energy Trading

Publication Data

> Akash Madhusudan, Hilder Vitor Lima Pereira, and Mustafa A. Mustafa. "Fully Privacy-Preserving Billing Models for Peer-to-Peer Energy Trading".

We are polishing the writing of this paper now. We will submit it to a journal as soon as it is finished.

Contributions

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

Fully Privacy-preserving Billing Models for Peer-to-Peer Electricity Trading Markets

Akash Madhusudan, Hilder VL Pereira, and Mustafa A. Mustafa

KU Leuven, ESAT/COSIC, Belgium, Universidade Estadual de Campinas, Brazil and University of Manchester
{akash.madhusudan}@esat.kuleuven.be

Abstract. Peer-to-Peer energy trading markets allow users to reduce their costs (when buying) or increase their revenues (when selling) electricity. In practice, there are often discrepancies between electricity volumes that are committed to an auction and the volumes that are actually consumed/injected. Recently, billing protocols that minimize such discrepancies have been proposed that utilize consumption-/production-specific information in order to be functional.

This paper proposes a novel fully privacy-preserving billing protocol that utilizes Fully Homomorphic Encryption (FHE) that prevents any third-party from learning sensitive consumption/production information of users, yet still takes account discrepancies between promised volumes and consumed volumes.

We utilize real-time consumption/production data over multiple billing periods and experimentally demonstrate the effectiveness and efficiency of our billing model.

1 Introduction

Peer-to-peer (P2P) electricity trading markets allow households to trade with each other through auction mechanisms, giving them more flexibility as opposed to trading only with their contracted supplier on fixed prices [5]. This flexibility brings opportunities for households to be more selective about: (1) when they trade their surplus electricity, and (2) the selected trading price when they devise their auction bids. The rational strategy while making these selections is to reduce their costs (when buying) or increasing their revenues (when selling) electricity [26]. Such markets also encourage local energy production, consumption and balancing, which also brings environmental benefits to local communities [23].

However, in practice there are often discrepancies between trading volumes that are committed at an auction (bid volumes), and the volumes actually consumed/injected by the participating households (actual smart meter readings). This happens because of the following factors: (1) volatility of the consumption/production patterns of the households [20], (2) uncertainty in the outputs of local renewable sources [15], and (3) the inaccuracy of prediction algorithms used by households to devise their auction bids [28]. To maximise the volumes of energy traded at the P2P markets, new billing models that minimise these discrepancies have been proposed [21].

These billing models provide ways to share the discrepancies between participating households in order to minimize penalties. However, these models (and P2P markets in general) require more fine-grain data (such as bid volume, meter readings and price per trading slot) to be shared with other households or the market/auction operator [5]. Such fine-grain data pose significant privacy risks [24]. Potentially, a malicious actor can deploy non-intrusive load monitoring techniques [7] and can easily infer sensitive information about the household occupants such as presence, activities, religion, health conditions, etc. [22].

There have already been attempts to protect households' data by designing privacy-preserving billing protocols for P2P energy trading markets. However, they have the following limitations. Solutions such as [3] do not take into account the aforementioned discrepancies and assume perfect fulfillment of promised bids. The ones that that provide privacy-preserving billings while taking the discrepancies into account [27, 16, 13] use partially homomorphic encryption, thus they rely on third parties to calculate the bills. In general, due to the nature of these protocols, the control over the user's data is given to the supplier. Furthermore, none of these solutions evaluate their protocols with real-time consumption/production data.

To address these limitations, this paper proposes a novel privacy-preserving billing protocol that utilizes fully homomorphic encryption to calculate encrypted bills which can only be decrypted by a participating household. More specifically, the novel contributions of this paper are:

- Utilizing fully homomorphic encryption in order to encrypt every household's data with their own key. This enables us to ensure that the control of a household's data is never given to any third party.
- We utilize the consumption data of 150 households, out of which 30 were prosumers and 120 were consumers. The prosumers were a mixture of households with wind turbines, solar panels, or both. We simulated the production of these renewable sources based on real-time weather conditions.

The rest of the paper is organized as follows: in Sect. 2 we discuss research that has already been done in the field of billings. Then, in Sect. 3 we discuss our system model,

threat model, design requirements, our previous non-privacy-preserving billing models, and the cryptographic primitives that we use to make these models privacy-preserving. Section 4 details our fully privacy-preseving billing model with universal cost split, followed by its evaluation in Sect. 5. Finally, Sect. 6 concludes this paper and lists some future work directions.

2 Related work

Privacy risks in P2P electricity trading markets (and in smart metering [18] in general) have already been highlighted [24]. There have already been attempts to reduce these risks. Below we first review the privacy-preserving billing for smart metering before focusing on the P2P electricity trading market.

Rial and Danezis [25] proposed a privacy-preserving protocol for time-of-use billing using fine-grained meter readings, which never leave the user households. Their protocol allows users to compute their bill locally and prove its correctness without disclosing any fine-grained consumption with the help of zero-knowledge proofs. Realising that even the final bill may leak users' private information, Danezis et al [9] has proposed to add some small amount of noise to the final bill, so that they can offer strong privacy guarantees. This is achieved by using differential privacy. One disadvantage of this method though is that 'noise' in this case corresponds to real money, which users must pay to preserve their privacy. To address this limitation, the authors proposed a complementary billing protocol that relies on cryptographic oblivious mechanism to support rebates. With the help of this additional protocol, users can deposit initial monetary values that can support positive and negative noise added to their bills, while ensuring that the funds they pay always cover the cost of their consumption.

Erkin and Tsudik [14] proposed privacy-preserving spatial and temporal aggregation of smart metering data. Their solution for temporary aggregation is suitable for private billing. Each smart meter generates random numbers equal to the number of time slots in a billing period. These random numbers are used in the encryption of each of the fine-grained metering data with the smart meter manufacturer's homomorphic public key. The aggregation is either computed at the supplier side or at the smart meter itself. Although none of the individual smart meter readings are leaked to the supplier or the manufacturer, the manufacturer also has access to users' monthly bill. A similar approach has also be taken by Jawurek et al. [17]. They proposed a privacy-preserving billing protocol that relies on zero-knowledge proofs based on Pedersen commitments performed by a plug-in privacy component placed into the communication link between the smart meter and supplier.

Danezis et al. [8] proposed complex non-linear billing protocols for smart meter readings that deploy multi-party computation (based on secret sharing techniques). The limitation of these protocols is that they rely on several semi-honest authorities. Similarly, Ababneh et al. [1] proposed a scheme for privacy-preserving electricity billing that deploys secret sharing and blinding techniques. In their proposed scheme, each smart meter blinds each of its meter readings and secretly shares them with its peers as well as it aggregates all the blinded readings for all the time slots per billing period and shares the aggregate value with a lead smart meter. The lead smart meter then verifies that both blinded aggregate values are the same and shares this value with the supplier. The supplier then unblinds the aggregate value and uses it to calculate the final user bill. The limitation of this scheme is its high communication cost as it requires many-to-many communication links amongst the smart meters themselves.

Inspired by the existing solutions for privacy-preserving billing for smart metering, there have already been attempts to provide billing solutions for P2P electricity markets that protect users' privacy.

Abidin et al. [2] proposed a simple billing protocol for P2P electricity trading market. Each smart meter calculates a fine-grained bill per time slot, and then masks it before sending the masked bill to a supplier. The masked values are selected such that they cancel each other when summed up in a billing period. Once the supplier gets all the masked values from a smart meter for the billing period, it aggregates them, which results in the monthly bill of the user (as the random numbers cancel each other). Our work is similar to their work because only the household/smart meter has access to the consumption/production data and the calculated bill. However, their work does not consider individual deviations of households in the bill calculation and assumes perfect fulfillment. In addition, since our bills are computed by a third-party server, changes to a billing model can be adopted more efficiently.

Thandi and Mustafa [27] proposed the first privacy-preserving billing solutions for P2P trading markets that takes into account discrepancies between what volumes smart meters commit and what volumes they finally deliver during a trading period. Their idea is these discrepancies are traded at the retail market. To calculate what volumes are traded at the retail and what at the P2P market, they deploy partial homomorphic encryption scheme. Hutu and Mustafa [16] improved the solutions proposed in [27] by proposing privacy-preserving billing protocols that can accommodate several (more complex) billing models for reducing users' bills by (more) fairly splitting the cost amongst all users though the deployment of weighting in the calculations of each user's individual contribution towards the overall discrepancy. All the calculations are computed using partial homomorphic encryption scheme.

Erdayandi et al. [13] deployed blockchain to store hashes of the committed volumes by the smart meters as well as the real meter readings. These readings are later used in the verification of the user bill. The final bill of consumers is calculated by prosumers and

vice versa using partial homomorphic encryption scheme. For example, for each billing period a consumer is matched with several prosumers, who receive encrypted data from the consumer and calculate the consumer's encrypted bill. The bill is then sent to the supplier, who can decrypt it and obtain the user's bill. Alqahtani and Mustafa [4] further improved the billing models by incorporating different zones, such that in-zones trades are encouraged, which by itself reduce user bills. These solutions deploy multi-party computation technique to ensure that the bills are calculated in a privacy-preserving manner.

In summary, existing solutions either assume perfect fulfillment of promised bids [3], or use partially homomorphic encryption schemes [27, 16, 13], thus they rely on third parties to calculate the bills, or multi-party computation technique, thus introduces too much communication overhead.

To address these limitations, we propose a novel privacy-preserving billing protocol for P2P electricity markets that utilizes fully homomorphic encryption to calculate encrypted bills, which can only be decrypted by a participating user.

3 Preliminaries

3.1 System Model

The system model used in this work consists of the following entities: *users*, *suppliers*, *grid operators* and *market operators*. Users are equipped with smart meters (SMs) who can measure the volume of electricity they consume/inject to the grid. Users who can only consume electricity are known as Consumers and ones who consume and produce electricity, e.g., by using RES (Renewable Energy Sources) are called Prosumers. Additionally, grid operators also perform aggregation required for our billing models, thus we use the terms grid operator and aggregator interchangeably.

The billing models that we propose involve communication between prosumers, consumers and suppliers. Prosumers and consumers live in households with inbuilt Home Energy Management Systems (HEMS) and Smart Meters (SM); additionally, Prosumers also have RES available to them in their household. Their functions and interactions are described as follows:

- Prosumers and Consumers: The entities who pay the bills derived by the proposed billing models are called prosumers and consumers. Prosumers are participants in the P2P energy trading markets that produce and consume electricity; whereas, consumers only do the latter.

- Smart Meters (SM) and Home Energy Management System (HEMS): These inbuilt devices provide real-time measurements of production/usage of the households they are installed in. Additionally, they also aid in facilitating P2P trading by performing functions and providing information necessary for our proposed billing models to work.
- Suppliers: In all our proposed billing models, the suppliers act as a fall-back options in case of any deviation caused by the P2P energy trading market. Trading with suppliers usually incurs a Retail Price (RP) that is higher than the Trading Price (TP) of P2P energy trading and is considered a penalty for the deviating users.
- Grid operator: The entities that manage the injection and output of electricity to/from the grid are known as grid operators. They need to ensure that the grid is always balanced in order to avoid any outages. In our system, we assume that grid operators also perform aggregation of inputs from households to calculate the total deviation in the system and run the billing models.
- Market Operator: The market operator takes the privacy-preserving offers/bids from the prosumers/consumers and runs a blind auction mechanism to match them and determine a trading price. Once its function is complete, only the participating prosumers and consumers know of their eligibility to participate in the P2P trading as this is done in a privacy-preserving way [2].

3.2 Threat Model and Assumptions

We assume the existence of honest-but-curious adversaries in case of our billing models. The grid operator and suppliers act according to the protocol but are curious about the information they receive. The prosumers and consumers follow the protocol specifications and pay the computed bill without altering it. We assume that the SMs are tamper-proof, and that the prosumers and consumers act rationally. All entities in our billing models are assumed to communicate over secure and authenticated communication channels. Finally, we assume that a household can either sell or buy electricity in a given timeslot; it cannot do both.

3.3 Design Requirements

We split the design requirements for our billing models into functional, security and privacy requirements and discuss them as follows:

Functional Requirements

The functional requirements of our billing models are as follows:

- **Consideration of deviations**: The billing models in this paper are designed to handle positive or negative deviations from all households and should be able to function in the presence of any arbitrary number of deviations for each time period.
- **Fairness**: The incurred penalties due to deviations should be split fairly amongst all the deviating entities.

Security and Privacy Requirements

The security and privacy requirements of our billing models are as follows:

- **Authenticity of consumption/production data**: The integrity and origin of all production/consumption specific data should be validated by the aggregator.
- **Non-repudiation of calculated bills**: The bills calculated by the aggregator should be authenticated as such.
- **Unlinkability of prosumers and consumers**: Any third party should not be able to link a pair of prosumer and consumer that trade with each other.
- **Confidentiality of billing data**: Any data that can classify consumption/production patterns of a household should not be accessible by any third party in plain. Only the encrypted version of this data should be available.

3.4 Billing Models

In this section, we briefly revisit the billing model with universal cost split proposed by Madhusudan et al. in [21]. We first start with the billing model for retail markets only (the status quo). We then briefly describe a P2P market with RM (the status quo) acting as a fall-back option. Finally, we present the billing model with universal cost split.

Billing Model for Retail Markets – the status quo

Algorithm 6 presents a billing model that is used in current retail markets. In this billing model, customers buy/sell their electricity only from/to suppliers they have a contract

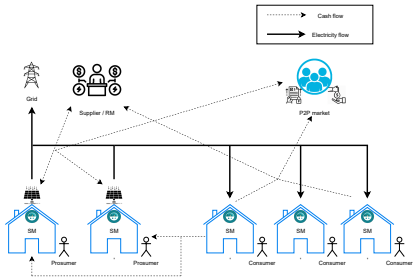


Figure 5.1: P2P market with RM as a fall-back option

with. The algorithm shown below takes as input the consumption and production of a customer in addition to the retail and feed-in tariff set by the market.

Algorithm 6 Billing Model for Retail Markets [21]

Input: Consumption (C_{dem}), production (P_{sup}), RP, FiT
Output: C bill, P reward, Supplier (S) balance

1: C bill = $C_{dem} \times$ RP
2: P reward = $P_{sup} \times$ FiT
3: S balance = C bill - P reward

As shown in Alg. 6, a consumer (C) is charged a retail price (RP) per KWh for the electricity they consume; similarly, a prosumer (P) gets paid a feed-in tariff (FiT) per KWh of electricity they inject to the grid. The balance of the supplier (S) is derived by the difference of their earnings through billing consumers and expenditure through paying prosumers.

P2P market with RM as Back-up

This P2P market uses the retail market as a backup in case there is a deviation in the volume of electricity that was committed to be consumed/supplied and the actual volume delivered. Each customer has an individual deviation ($InDev_x$) where x=p for a prosumer and x=c for a consumer. Figure 5.1 depicts how such an interaction would look like. Briefly, P2P trading markets run as follows:

- Consumers/Prosumers submit bids/offers for electricity;
- A market operator (e.g., trading platform) clears the P2P market. It determines the total cleared volume of electricity at the market, the market clearance price and the consumers/prosumers whose bids/offers have been accepted.

Consumers/Prosumers whose bids have not been accepted use the RM to meet their needs.

- Finally, meter readings from the smart meters of consumers/prosumers is used to calculate their bills/rewards. If the data from the smart meters of prosumers/consumers whose bids have been accepted is not the same as their respective committed value in the bid/offer, the difference (deviation) is compensated by the RM. This could come with an extra cost for some of the prosumers/consumers.

In all future billing models, we do not explicitly list the supplier balance as that is not the main focus of this work and it is straightforward to calculate it. The balance of a supplier (S_{bal}) is always the difference between their income S_{inc} = demand x RP and their expenditure S_{exp} = supply x FiT.

The cost associated with compensating the deviations of the individual prosumers/-consumers will have to be distributed amongst all (some) of the prosumers/consumers to reduce the dependency on retail market as a fallback. There could be different mechanisms to share these costs amongst prosumers/consumers. Madhusudan et al. [21] propose three different methods of how these costs could be shared: individual, social and universal cost split. In this paper, we focus explicitly on universal cost split.

Billing model with universal cost split

This billing model aggregates all the individual deviations of P2P market customers (prosumers and consumers) and the cost of this resulting total deviation is socially split amongst the customers whose individual deviation is in the same direction as the total deviation. This is described below and shown in Alg. 7.

All individual deviations are aggregated to calculate the total deviation of the P2P market. If the total deviation is equal to zero, all prosumers (consumers) sell (buy) at TP, regardless of their individual deviations. However, if the total deviation is positive (total supply is more than the total demand), then all consumers buy all their consumed volume at the TP, regardless of their individual deviation, whereas prosumers split the reward calculated based on the total deviation. Similarly, if the total deviation is negative, then all prosumers sell all their supplied volume at the TP, regardless of their individual deviation, whereas the consumers split the cost calculated based on the total deviation.

Algorithm 7 Billing Model with Universal Cost Split [21]

Input: Consumption, production, deviation
Output: Customer bill, Prosumer reward

if bid Accepted **then**
  if TD = 0 **then**
    $P2P_c$ bill = $C_{dem}^{P2P} \times$ TP
    $P2P_p$ reward = $P_{sup}^{P2P} \times$ TP
  if TD < 0 **then**
    $P2P_p$ reward = $P_{sup}^{P2P} \times$ TP
    if $\text{InDev}_c = 0$ or $\text{InDev}_c < 0$ **then**
      $P2P_c$ bill = $C_{dem}^{P2P} \times$ TP
    if $\text{InDev}_c > 0$ **then**
      $P2P_c$ bill = $(C_{dem}^{P2P} - \frac{TD}{P2P_n^c}) \times$ TP + $\frac{TD}{P2P_n^c} \times$ RP
  if TD > 0 **then**
    $P2P_c$ bill = $C_{dem}^{P2P} \times$ TP
    if $\text{InDev}_p = 0$ or $\text{InDev}_p < 0$ **then**
      $P2P_p$ reward = $P_{sup}^{P2P} \times$ TP
    if $\text{InDev}_p > 0$ **then**
      $P2P_p$ reward = $(P_{sup}^{P2P} - \frac{TD}{P2P_n^p}) \times$ TP + $\frac{TD}{P2P_n^p} \times$ FiT
if bid not accepted **then**
  goto Algorithm 1

3.5 Cryptographic Primitives

Fully homomorphic encryption

Fully homomorphic encryption (FHE) is a powerful tool that allows operations to be perfomed on encrypted data. In more detail, a client generates a ciphertext c encrypting some data m under a public key pk. Then, a server holding c and pk, but not the corresponding secret key sk, can choose any computable function f and generate a new ciphertext c' that encrypts $f(m)$. We call this step the "homomorphic computation" and say that "f was evaluated homomorphically". Finally, the client can download c', decrypt it using sk, and obtain $f(m)$. Therefore, FHE allows one to securely outsource computation to an untrusted party.

Now, we present in high level the relevant operations that we use in our billing model.

- $\mathsf{HE.paramGen}(\lambda, p)$: given the security parameter λ and the plaintext precision p, generates parameters param, which include the *ring dimension* N, the *ciphertext modulus* Q, and the number of slots S.

- $\mathsf{HE.keyGen}(\mathsf{param})$: generates the encryption key pk, the decryption key sk, and also the relinearization key rlk which is used two multiply two ciphertexts.
- $\mathsf{HE.enc}(m(X))$: outputs a ciphertext $\mathbf{c}$ encrypting $m(X)$. We simplify the notation by saying that $\mathbf{c}$ encrypts the corresponding vector $\mathbf{u}$ encoded by $m(X)$ and write $\mathbf{c} = \mathsf{Enc}(u_1, ..., u_S)$.
- $\mathsf{HE.dec}(\mathbf{c})$: decrypts $\mathbf{c}$ and returns the message $m(X)$. We simplify the notation by including the decoding into the decryption and considering that the output is corresponding vector $\mathbf{u}$ encoded by $m(X)$.
- $\mathsf{HE.add}(\mathbf{c}_0, \mathbf{c}_1)$: consider that $\mathbf{c}_0$ and $\mathbf{c}_1$ encrypt vectors $\mathbf{u}$ and $\mathbf{v}$. This homomorphic operation outputs $\mathbf{c} = \mathsf{Enc}(\mathbf{w})$ where $\mathbf{w}[i] = \mathbf{u}[i] + \mathbf{v}[i] \mod p$.
- $\mathsf{HE.mult}(\mathbf{c}_0, \mathbf{c}_1, \mathsf{rlk})$: consider that $\mathbf{c}_0$ and $\mathbf{c}_1$ encrypt vectors $\mathbf{u}$ and $\mathbf{v}$. This homomorphic operation outputs $\mathbf{c} = \mathsf{Enc}(\mathbf{w})$ where $\mathbf{w}[i] = \mathbf{u}[i] \cdot \mathbf{v}[i] \mod p$. Notice that it requires the relinearization key.

Pseudo-random zero sharing

In this section, we present a simplified version of the Pseudo-random zero sharing (PRZS) protocol from Appendix A of [19]. Here, we do not need commitment schemes in the setup phase, since we assume the semi-honest model, i.e., all parties follow the protocol. This protocol allows k parties to generate random-looking shares that add up to zero. Those shares can then be used to mask values that the users want to send to other users. The PRZS is defined as follows:

- $\mathsf{Setup}(\lambda, k)$: each user $1 \leq u \leq k$ generates PRF keys $K_{u,v}$, for $1 \leq v \leq k$ and $u \neq v$ and sends $K_{u,v}$ to user v. The keys are long enough so that the PRF achieves λ bits of security.
- $\mathsf{GenShare}(u, \mathsf{count})$: define $a_i := \mathsf{PRF}_{K_{u,i}}(\mathsf{count})$ and $b_i := \mathsf{PRF}_{K_{i,u}}(\mathsf{count})$ for $i \neq u$. Define a as the sum of all a_i's. b as the sum of all b_i's. The user u's share as $s_u = a - b$. Add one to count.

It is clear that $\sum_{u=1}^{k} s_u = 0$. Also, each s_u is pseudorandom.

4 Privacy-enhanced billing models

This section introduces a privacy-preserving variant of the universal cost-split billing model that collectively enables customers to settle their electricity bills in a fully privacy-preserving way while reducing the dependency on retail markets.

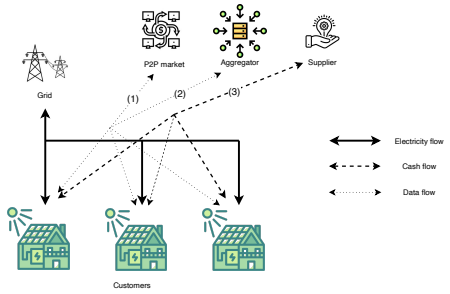


Figure 5.2: Overview of our privacy-preserving billing model

4.1 Overview

Before diving into the details, we give an overview of how this model works. This model is depicted in Fig. 5.2.

In order to participate in the P2P market, each customer (consumer or prosumer) participates in the auction held on the trading platform [as shown in step (1)]. The customers that are selected to participate in P2P trading send their encrypted information about consumption, production and deviation to the grid operator [as shown in step (2)]. The grid operator then runs the privacy-preserving billing model on their side, i.e., they calculate the bill/reward of the customer without requiring any plaintext information. Consequently, they output the encrypted bill/reward, which is then sent to the customer. Upon receiving the encrypted bill/reward, the customer decrypts it with their secret key to reveal how much they owe (resp. owed) to (resp. by) the supplier. The customer then performs the payment or receives it as required [as shown in step (3)].

4.2 Setup Phase

Before we can start utilizing the billing models, we need to perform the setup necessary to generate the required cryptographic keys for each cryptographic primitive that we utilize, i.e., pseudo-random zero sharing [19] and FHE.

As discussed in Sec. 3.5, each of these primitives has a setup phase that has to be executed *once* for a set of participants utilizing our billing models in order to receive their keys. After generating these keys, each participant can use them whenever they partake in P2P electricity trading.

Specifically, for Pseudo-Random zero sharing the participant needs to execute the $\mathsf{Setup}(\lambda, k)$ where λ is the desired security parameter and k is the number of participants.

Note that this only needs to be done once and the output of this setup are the PRF keys required to generate masking shares for the individual deviations.

Additionally, for FHE the participant needs to execute $\mathsf{HE.paramGen}(\lambda, p)$ and $\mathsf{HE.keyGen}(\mathsf{param})$ in order to generate the necessary keypair for encrypting and decrypting their deviations, consumptions and productions. This also needs to be done only once.

By the end of this setup phase, we assume that all participants are in possession of the required cryptographic keys.

4.3 Client Setup for Billing Model with Universal Cost Split

The client setup is depicted in Alg. 8. During the client setup, the smart meters on the client's side collect the energy consumption and/or production corresponding to each time slot (e.g., every 15 minutes). Using this data, the smart meter computes the deviations as compared to their committed values in the P2P market. Ideally, the smart meters can just encrypt each of these measurements in a different ciphertext and send it to the server (grid operator) to be processed. However, that would sub-utilize the FHE ciphertexts, since they can encrypt a vector. Hence, for the sake of efficiency, the smart meters first collect several measurements, populate a vector, encrypt it, and only then send the ciphertexts to the grid operator.

Moreover, to simplify the homomorphic computation performed on the server side, the smart meter also encrypts the sign of each deviation. More specifically, the grid operator has to compute values that depend on whether the deviation is negative or positive. Thus, given a ciphertext encrypting the deviation, the grid operator would have to compute a homomorphic comparison, which is rather expensive. However, if the grid operator already has a ciphertext encrypting the sign, then the homomorphic comparison is replaced by two homomorphic multiplications, which is more computationally efficient. The way Alg. 8 proceeds is as follows:

- Lines 5-9: The smart meter collects the measurements of consumption, production, individual deviations and signs of deviations for each time slot.
- Lines 10-13: The smart meter sets a negative or positive bit for each individiual deviation, i.e., whether its under-consumption/production or over-consumption/production. Additionally, each household derives their masking s_t and calculates the masked individual deviation $\mathbf{md}[t]$ per time slot t.
- Lines 14-18: At the end of each time slot, the smart meter homomorphically encrypts these vectors and sends them to the grid operator.

Algorithm 8 Client Data Collection

Input: Number of slots S. Client's secret key sk. Global Count *count*.

Output: Ciphertexts $\mathbf{c}_c$, $\mathbf{c}_d$, $\mathbf{c}_s$, and $\mathbf{c}_b$ encrypting, respectively, vectors of consumptions, individual devitations, supplies, and bits.

$\mathbf{c} = (0,...,0) \in \mathbb{Z}^S$ → Vector of consumptions
$\mathbf{s} = (0,...,0) \in \mathbb{Z}^S$ → Vector of supplies
$\mathbf{d} = (0,...,0) \in \mathbb{Z}^S$ → Individual deviations
$\mathbf{md} = (0,...,0) \in \mathbb{Z}^S$ → Masked individual deviations
$\mathbf{b} = (0,...,0) \in \mathbb{Z}^S$ → Signs of deviations
for each time slot $0 \leq t < S$ **do**
    $\mathbf{c}[t] = \mathsf{P2P}_{i,t}^{dem}$
    $\mathbf{s}[t] = \mathsf{P2P}_{i,t}^{sup}$
    $\mathbf{d}[t] = \mathsf{InDev}_{i,t}$
    if $\mathsf{InDev}_{i,t} \leq 0$ **then**
        $\mathbf{b}[t] = 1$
    s_t = GenShare(id, *count*) $\in \mathbb{Z}_{2^\ell}$
    $\mathbf{md}[t] = s_t + \mathbf{d}[t] \bmod 2^\ell$
$\mathbf{c}_c = \mathsf{HE.enc}_{\mathsf{sk}}(\mathbf{c})$
$\mathbf{c}_s = \mathsf{HE.enc}_{\mathsf{sk}}(\mathbf{s})$
$\mathbf{c}_d = \mathsf{HE.enc}_{\mathsf{sk}}(\mathbf{d})$
$\mathbf{c}_b = \mathsf{HE.enc}_{\mathsf{sk}}(\mathbf{b})$
Send $\mathbf{c}_c, \mathbf{c}_s, \mathbf{c}_d, \mathbf{c}_b, \mathbf{md}$ to the server

4.4 Server's Billing Model with Universal Cost Split

Upon receiving the encrypted data from the smart meters, the server (grid operator) executes a homomorphic version of Algorithm 7, which we describe in detail in Algorithm 9. As each ciphertext encrypts a vector of dimension S, where S denotes the total measurements (number of slots) in the vector, the homomorphic version is actually equivalent to running Algorithm 7 in parallel for S different inputs in a privacy-preserving way.

All the branches, i.e., if/else statements of the original algorithm are computed homomorphically based on the fact that the code "`if B then f(x) else g(x)`", where `B` is a Boolean value, is equivalent to "`B*f(x) + (1 - B)*g(x)`". Hence, if `B` is equal to 1, then we end up with $f(x)$. Otherwise, $f(x)$ is multiplied by zero and we have $g(x)$.

For the branches that depend on the individual deviations, note that the smart meter uploads encryptions of bits that indicate whether $\mathrm{InDev}_c \leq 0$ or not. Thus, we homomorphically compute the bill and the reward corresponding to both cases, i.e., $\mathrm{InDev}_c \leq 0$ or $\mathrm{InDev}_c > 0$.

Finally, we multiply those bills and rewards by the ciphertexts $\mathbf{c}_b$ and $\mathbf{c}_b = 1 - \mathbf{c}_b$, with the bits corresponding to signs of the individual deviations, so that only the correct bills and rewards are multiplied by one, while the others are nullified.

The same strategy is used for the if/elses' of Algorithm 7 that depend on the total deviation. Since this value is in clear for the server (because it is obtained via an additive secret sharing protocol), the server can produce, by its own, the bits corresponding to the three following cases: $totDev_i$ is less than zero, it is equal to zero, or it is greater than zero. Then, we homomorphically compute the bills and rewards corresponding to each of these three cases, and multiply by those bits so that we only keep the correct bills and rewards.

For example, if for the fourth time slot, the total deviation ($\mathbf{w}$) is zero, then we have $\mathbf{w}^{(0)}[4] = 1$ while $\mathbf{w}^{(<0)}[4] = \mathbf{w}^{(>0)}[4] = 0$. Then, as shown in lines 17 and 18 of Alg. 9, we compute the bills/rewards using the formulas of the three possible cases, i.e.: $\mathbf{c}_{bill}^{(0)}$ has in its S slots the bills computed as if all the total deviations were 0, whereas $\mathbf{c}_{bill}^{(<0)}$ encrypts bills considering negative total deviations, and $\mathbf{c}_{bill}^{(>0)}$ encrypts bills considering positive total deviations. Hence, at the end of the algorithm, when we compute

$$\mathbf{c}_{bill} = \mathbf{c}_{bill}^{(0)} \cdot \mathbf{w}^{(0)} + \mathbf{c}_{bill}^{(<0)} \cdot \mathbf{w}^{(<0)} + \mathbf{c}_{bill}^{(>0)} \cdot \mathbf{w}^{(>0)}$$

the fourth slot of the message encrypted by $\mathbf{c}_{bill}$ will encrypt the bill corresponding to the $\mathsf{TD}_4 = 0$. The same argument applies to all the other slots and to the rewards.

4.5 Bill settlement

After the successful execution of Alg. 9 the client receives $\mathbf{c}_{bill}$, $\mathbf{c}_{reward}$, or both via an authenticated channel from the grid operator.

Upon receiving these encrypted bill/reward, they can decrypt them using the private key that they generated during the setup phase (see Sec 4.2). Once decrypted, the client can pay the required amount to the supplier, or request the reward amount from the supplier. Along with the payment, the client also sends $\mathbf{c}_{bill}$ and/or $\mathbf{c}_{reward}$ along with a zero-knowledge proof of valid decryption.

5 Evaluation

5.1 Security Analysis

The security and privacy requirements of our billing models (as listed in Sec. 3.2) are **(1). Authenticity of consumption/production data, (2). Non-repudiation of calculated bills, (3). Unlinkability of prosumers and consumers,** and **(4). Confidentiality of billing data.**

Authenticity of consumption/production data is guaranteed by our assumption that SM's are tamper-proof. They collect data about the volume of electricity being consumed/produced without any alteration, perform the necessary operations and communicate this to the grid operator. Additionally, the provided data is also signed by the SM for non-repudiation and authenticity.

Non-repudiation of calculated bills is guaranteed due to the requirement for grid operators to sign all computed bills/rewards before sending them to the respective households. We assume that the signing algorithm guarantees the properties of *unforgeability*, *verifiability*; hence, we can guarantee the fulfillment of this property.

Unlinkability of prosumers and consumers is guaranteed by the protocol used by the market operator for running the privacy-preserving P2P auction in order to match trading bids and offers from prosumers and consumers and derive a trading price per timeslot. Protocols such as the one proposed by Botelho Da Gama [6] guarantee unlinkability of prosumers and consumers that trade with each other.

Confidentiality of billing data is guaranteed by the primitives that we use for our billing model. The utilized pseudo-random zero sharing [19] is formally proven to provide the property of *pseudo-randomness*, i.e., the masked individual deviations calculated in Alg. 8 are indistinguishable from truly random values. Since we assume passive adversaries, they cannot gain any information about the individual deviations of the households from the masked values. Additionally, our utilized FHE algorithms for calculating the bills is formally proven to be IND-CPA secure, i.e., the encrypted volumes and deviations do not leak any information about the actual volumes and deviations of each household.

Hence, using these two primitives in conjuction to calculate the bills/rewards for households does not leak any private information and guarantees the fulfillment of this property.

5.2 Theoretical analysis

Our protocol composes two different primitives, namely, pseudo-random zero sharing (PRZS) and fully homomorphic encryption (FHE). Since PRZS only uses symmetric encryption, it is time complexity is negligible compared to the cost of FHE. The only important point to notice about PRZS is that its communication cost is quadratic in the number of users, thus, in the setup phase, if we use groups of k users, each user has to send $k-1$ symmetric keys. Since each of those keys has linear size in the security parameter λ, the communication cost per user is $O(k\lambda)$.

An important metric in FHE schemes is the depth of the homomorphic computation to be evaluated, which is the number of ciphertext-ciphertexts multiplications that have to be performed in sequence. By inspecting Algorithm 9, we can see that the depth is only 1. Indeed, one can see that the variable $\mathbf{c}_{bill}^{(<0)}$ is the one with the longest chain of multiplications, but it turns out that it is of the form

$$\mathbf{c}_{bill}^{(<0)} = \mathbf{c}_b \cdot \mathbf{c}_c \cdot \mathbf{u} + \mathbf{c}'_b \cdot \mathbf{c}_c \cdot \mathbf{u} + \mathbf{c}'_b \cdot \mathbf{p}$$

where $\mathbf{u}$ and $\mathbf{p}$ are plaintexts and $\mathbf{c}_b, \mathbf{c}_c$, and $\mathbf{c}'_b$ are ciphertexts.

Since the multiplicative depth is only 1, the client can instantiate the FHE scheme with a very small ciphertext modulus, for example, $Q = 2^{64}$. For generality, we consider then $Q = O(1)$, i.e., a constant. For such small modulus, we can obtain λ bits of security by setting $N = \Theta(\lambda)$ with very small hidden constant. For instance, for $\lambda = 128$, it is possible to choose $N = 1024$. Hence, the size of each ciphertext is $2N \log Q = O(\lambda)$.

By analyzing Algorithm 8, we see that the only overhead the client has compared to the original setup algorithm with no privacy guarantee is the sequence of 4 encryptions at the end, which just cost time $O(\lambda \log \lambda)$. Also, the communication cost is just $O(\lambda)$, since the client just has to upload those 4 ciphertexts to the server. However, notice that since each ciphertext encrypts $S = \Theta(\lambda)$ messages, the overhead compared to the plaintext version is just $O(1)$.

To analyze the cost of Algorithm 9, i.e., how the server process the queries, we can focus on the homomorphic multiplications, as they are by far more costly than the other operations. We can see that the number of multiplications is constant and, actually, very close to the number of multiplications performed by the original algorithm, which runs in clear, thus, we can measure the overhead by estimating the cost of one homomorphic multiplication divided by the cost of one multiplication in clear. The latter just costs one instruction. But one homomorphic multiplication is performed with two or three degree-N polynomial multiplications, which can be performed in time $O(N \log N)$ using Fast Fourier Transform. Again, remember that each homomorphic operation acts on $S = \Theta(N)$ slots in parallel, thus, the amortized cost is $O(\log N)$. Because we can set $N = O(\lambda)$, the overhead of Algorithm 9 is then just $O(\log \lambda)$.

Summarizing, to add privacy guarantees to the algorithms, we have almost no overhead in the communication, i.e., $O(1)$, and a very small overhead for the server running time, i.e., $O(\log \lambda)$, where λ is the security parameter and is typically set as 128. In Section 5.3, we show concrete numbers obtained with some experiments to measure the overhead in practice instead of only asymptotically.

5.3 Simulations

Use-case

To empirically evaluate our proposed solution, we simulate and analyse the following use-case.

There is a local community of 150 households, of which 120 are consumers and 30 prosumers. The RES owned by the prosumers are as follows: 20 prosumers have photovoltaic (PV), 5 consumers have wind turbines (WTs), while the remaining 5 prosumers have both PVs and WTs.

For each trading slot, each of the households submits a bid/offer to a P2P trading market. The market runs a double auction to select the winning bids/offers and to determine the trading price. The households whose bids are accepted are expected to fulfil their obligation (committed via their bids) by demanding/supplying the corresponding volumes of electricity. After the trading period, real data from the smart meters are used to calculate individual bills. Households whose bids are rejected trade on the retail market, while households whose bids are accepted trade on the P2P market (and the retail market if their smart meter data deviates from their committed volume of electricity in their bids/offers).

The grid operator calculates the bills of households. Households could choose to be billed daily, weekly or monthly.

Data Generation

We use real as well as simulated data in our experiments. The type of data we use are: consumption data for 150 households, PV generation data for 25 households and WT generation data for 10 households; committed volumes of electricity at the P2P market by the households whose bids/offers are accepted; trading price at the P2P market as well as the prices at the retail market.

We gather/simulate data for the period 1 – 7 July 2020 for the Brussels region, using 1 hour long trading slots.

Real consumption data collected from 61 Belgian households as part of the Linear project [10] are used as as example data as well as a base for generating the consumption data for further 89 households. To this data generation, we follow the following approach for each household. We randomly select one of the 61 households and multiply their consumption data with a randomly selected coefficient from the range [0.9 – 1.1]. In summary, for the consumption data of households, we use real data from 61 real households and synthetic data for 89 households derived from real data.

For the PV generation data, for each of the households with PVs, we randomly choose a PV capacity from the set [2.3, 3.6, 4.7] KWp. Related to the strength of the solar radiation at a particular time slot, for each PV, we set a PV output equal to the percentage of its capacity. For example, on 7 July 2023 at 13:15 CEST in Belgium, PVs with total capacity of 7.677,34 MWp generated 5.658,64 MW energy, resulting in a conversion coefficient of 0.737 [11]. To incorporate further deviations in their outcome, we have further multiplied each PV outcome by a coefficient randomly chosen from the range [0.9 – 1.1].

Similarly, for the WT generation data, for each of the households with WTs, we randomly chose a WT capacity from the following set [1.0, 1.5, 2.0] KWp. Related to the strength of the wind at a particular time slot, for each WT, we set an output equal to the percentage of its capacity. For example, on 7 July 2023 at 13:15 CEST in Belgium, WTs with total capacity of 2.525,9 MWp generated only 180,06 MW energy, resulting in a conversion coefficient of 0.0712 [12]. To incorporate further deviations in their outcome, we have further multiplied each WT outcome by a coefficient randomly chosen from the range [0.9 – 1.1].

The real demand/supply data of prosumers is then calculated by just subtracting their PV/WTs generation data from their consumption/supply data per each time slot.

We generate the predicted volume of each household from their consumption/supply data. We assume that 10% of households have a perfect prediction algorithm; 60% – good, 20% – not so good, and 10% – quite bad. We randomly select 10% of the households and make their predicted volume equal to their actual consumption/supply data. We randomly select 60% of the remaining households whose predicted volume deviates from their actual consumption/generation profile with [1–10%]. We randomly select 20% of the remaining households whose predicted volume deviates from their actual consumption/generation profile with [11-20%]. The predictions of the remaining 10% of the households deviate a lot from their actual consumption/supply data [21-100%].

We then generate the asking price between 5 and 20 cents. For people who want to sell electricity, we select a price from the range [6–20]. The higher volume they want to sell, the lower the asking price we set, and vice versa. For people who want to buy electricity, the more they want to buy, the higher the price they offer.

The WT data used is available here: https://github.com/PeijieZ/WindpowerlibTurbine-model.

Load profile data used is available here: https://github.com/PeijieZ/Load-profile-generation.

Bid/offer data, deviations, and trading price are available here: https://github.com/PeijieZ/Billing-Models-for-Electricity-Trading-Markets.

Simulation Results

We have also performed simulations to demonstrate the feasibility of our proposed solution in a real-world P2P market setting. Our simulations are split into two parts: computational cost and communication overhead.

Computational cost We split the computational cost into two parts: cost per trading slot and cost per billing period.

Our experiments were run one at a time on a near-idle PC; i.e., no multi threading. The machine we ran our simulations on had the following specifications: 13th Gen Intel® Core™ i7-13700Hwe @ 5 GHz and 16 GB RAM.

We split our experiments based on the number of participating households, and various billing periods. We obtained the following results:

- As shown in Fig. 5.3, the computation time of bill/reward per household remains almost the same, i.e., ~ 2.3 seconds, across the different number of households and lengths of billing period. This is because all the required data for calculating the bill/reward per household can be packed efficiently to be computed in FHE.
- Figure 5.4 depicts the computation times per billing period for 50 households. For each billing period, the minimum time required and the average time required remains the same, i.e., ~ 1.8 seconds and ~ 2.3 seconds respectively. However, the maximum computation time per billing period differs, with as high as 2.8 seconds for day billings to 2.6 seconds for month billing.

6 Conclusions and Future Work

This paper presented a fully privacy-preserving billing model based on universal cost-split that handles discrepancies between committed and actually consumed/injected

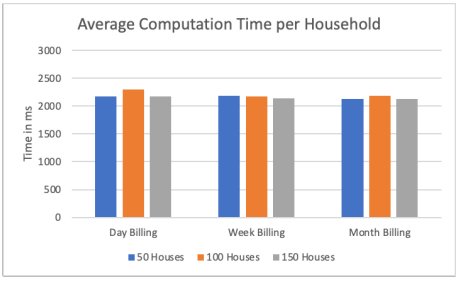


Figure 5.3: Average computation time for all households and billing periods

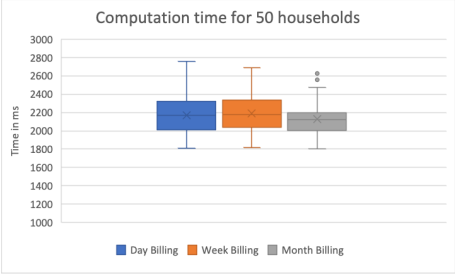


Figure 5.4: Computation times per billing period for 50 households

volumes of electricity. Our billing model is capable of calculating bills/rewards for all participating households with varying billing periods within ~ 2.5 seconds, showing its feasibility when deployed in a real-world setting.

As future work, we plan to execute the simulations of our billing model in a completely distributed fashion, i.e., the server and clients situated in geographically different locations. Additionally, we currently operate on the assumption that our adversary is honest but curious (or passive). While this fits our use case, as future work we plan to formally prove our billing protocol to be secure in the presence of an actively malicious adversary that can take any arbitrary action to sabotage the execution of the protocol.

References

[1] Mohammed Ababneh, Kartick Kolachala, and Roopa Vishwanathan. Private and secure smart meter billing. In *Proceedings of the 8th ACM on Cyber-Physical System Security Workshop*, pages 15–25, 2022.

[2] Aysajan Abidin, Abdelrahaman Aly, Sara Cleemput, and Mustafa A Mustafa. An

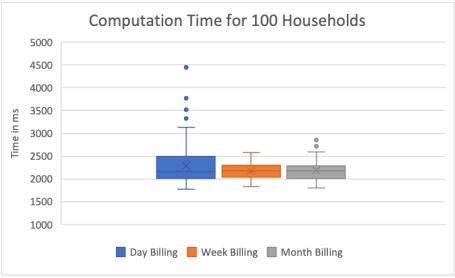


Figure 5.5: Computation times per billing period for 100 households

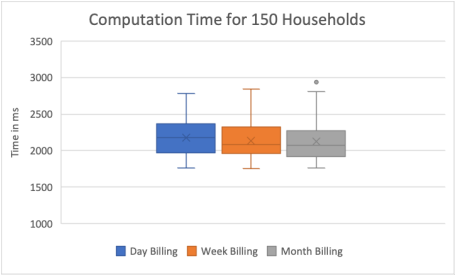


Figure 5.6: Computation times per billing period for 150 households

MPC-based privacy-preserving protocol for a local electricity trading market. In *Int. Conf. on Cryptology and Network Security*, pages 615–625, 2016.

[3] Aysajan Abidin, Abdelrahaman Aly, Sara Cleemput, and Mustafa A Mustafa. Secure and privacy-friendly local electricity trading and billing in smart grid. *arXiv preprint arXiv:1801.08354*, 2018.

[4] Eman Alqahtani and Mustafa A Mustafa. Zone-based privacy-preserving billing for local energy market based on multiparty computation. In *2023 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–7. IEEE, 2023.

[5] Timothy Capper, Anna Gorbatcheva, Mustafa A Mustafa, Mohamed Bahloul, Jan Marc Schwidtal, Ruzanna Chitchyan, Merlinda Andoni, Valentin Robu, Mehdi Montakhabi, Ian J Scott, et al. Peer-to-peer, community self-consumption, and transactive energy: A systematic literature review of local energy market models. *Renewable and Sustainable Energy Reviews*, 162:112403, 2022.

[6] Mariana Botelho da Gama, John Cartlidge, Antigoni Polychroniadou, Nigel P. Smart, and Younes Talibi Alaoui. Kicking-the-bucket: Fast privacy-preserving trading using buckets. In *Financial Cryptography and Data Security: 26th*

International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers, page 20–37, Berlin, Heidelberg, 2022. Springer-Verlag.

[7] Shuang Dai, Fanlin Meng, Qian Wang, and Xizhong Chen. Dp2-nilm: A distributed and privacy-preserving framework for non-intrusive load monitoring. *Renewable and Sustainable Energy Reviews*, 191:114091, 2024.

[8] George Danezis, Cédric Fournet, Markulf Kohlweiss, and Santiago Zanella-Béguelin. Smart meter aggregation via secret-sharing. In *Proceedings of the first ACM workshop on Smart energy grid security*, pages 75–80, 2013.

[9] George Danezis, Markulf Kohlweiss, and Alfredo Rial. Differentially private billing with rebates. In *Information Hiding: 13th International Conference, IH 2011, Prague, Czech Republic, May 18-20, 2011, Revised Selected Papers 13*, pages 148–162. Springer, 2011.

[10] R. D'hulst, W. Labeeuw, B. Beusen, S. Claessens, G. Deconinck, and K. Vanthournout. Demand response flexibility and flexibility potential of residential smart appliances: Experiences from large pilot test in belgium. *Applied Energy*, 155:79–90, 2015.

[11] Elia. Solar power generation. https://www.elia.be/en/grid-data/power-generation/solar-pv-power-generation-data, 2024.

[12] Elia. Wind power generation. https://www.elia.be/en/grid-data/power-generation/wind-power-generation, 2024.

[13] Kamil Erdayandi, Lucas C Cordeiro, and Mustafa A Mustafa. A privacy-preserving and accountable billing protocol for peer-to-peer energy trading markets. In *2023 International Conference on Smart Energy Systems and Technologies (SEST)*, pages 1–6. IEEE, 2023.

[14] Zekeriya Erkin and Gene Tsudik. Private computation of spatial and temporal power consumption with smart meters. In *Applied Cryptography and Network Security: 10th International Conference, ACNS 2012, Singapore, June 26-29, 2012. Proceedings 10*, pages 561–577. Springer, 2012.

[15] Kazi Nazmul Hasan, Robin Preece, and Jovica V Milanović. Existing approaches and trends in uncertainty modelling and probabilistic stability analysis of power systems with renewable generation. *Renewable and Sustainable Energy Reviews*, 101:168–180, 2019.

[16] Andrei Hutu and Mustafa A Mustafa. Privacy preserving billing in local energy markets with imperfect bid-offer fulfillment. In *2023 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–9. IEEE, 2023.

[17] Marek Jawurek, Martin Johns, and Konrad Rieck. Smart metering de-pseudonymization. In *27th annual conf. on computer security applications*, pages 227–236, 2011.

[18] Georgios Kalogridis, Mahesh Sooriyabandara, Zhong Fan, and Mustafa A Mustafa. Toward unified security and privacy protection for smart meter networks. *IEEE Systems Journal*, 8(2):641–654, 2013.

[19] Marcel Keller, Dragos Rotaru, Nigel P. Smart, and Tim Wood. Reducing communication channels in mpc. In Dario Catalano and Roberto De Prisco, editors, *Security and Cryptography for Networks*, pages 181–199, Cham, 2018. Springer International Publishing.

[20] Weicong Kong, Zhao Yang Dong, David J Hill, Fengji Luo, and Yan Xu. Short-term residential load forecasting based on resident behaviour learning. *IEEE Transactions on power systems*, 33(1):1087–1088, 2017.

[21] Akash Madhusudan, Fairouz Zobiri, and Mustafa A. Mustafa. Billing models for peer-to-peer electricity trading markets with imperfect bid-offer fulfillment. In *2022 IEEE International Smart Cities Conference (ISC2)*, pages 1–7, 2022.

[22] Eoghan McKenna, Ian Richardson, and Murray Thomson. Smart meter data: Balancing consumer privacy concerns with legitimate applications. *Energy Policy*, 41:807–814, 2012.

[23] Mehdi Montakhabi, Shenja van der Graaf, and Mustafa A Mustafa. Valuing the value: An affordances perspective on new models in the electricity market. *Energy Research & Social Science*, 96:102902, 2023.

[24] Mustafa A. Mustafa, Sara Cleemput, and Aysajan Abidin. A local electricity trading market: Security analysis. In *2016 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–6, 2016.

[25] Alfredo Rial and George Danezis. Privacy-preserving smart metering. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*, pages 49–60, 2011.

[26] Jan Marc Schwidtal, Proadpran Piccini, Matteo Troncia, Ruzanna Chitchyan, Mehdi Montakhabi, Christina Francis, Anna Gorbatcheva, Timothy Capper, Mustafa A Mustafa, Merlinda Andoni, et al. Emerging business models in local energy markets: A systematic review of peer-to-peer, community self-consumption, and transactive energy models. *Renewable and Sustainable Energy Reviews*, 179:113273, 2023.

[27] Robbie Thandi and Mustafa A Mustafa. Privacy-enhancing settlements protocol in peer-to-peer energy trading markets. In *2022 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pages 1–5. IEEE, 2022.

[28] Yingjie Wang, Fairouz Zobiri, Mustafa A Mustafa, James Nightingale, and Geert Deconinck. Consumption prediction with privacy concern: Application and evaluation of federated learning. *Sustainable Energy, Grids and Networks*, 38:101248, 2024.

Algorithm 9 Client Bill Calculation (by server)

Input: Number of slots S. Ciphertexts $\mathbf{c}_c$, $\mathbf{c}_d$, $\mathbf{c}_s$, and $\mathbf{c}_b$ encrypting, respectively, vectors of consumptions, indivial devitations, supplies, and bits. Vectors $\mathbf{md}_u$, for $1 \leq u \leq k$, containing the masked individual deviations of each user u.

Output: Ciphertexts $\mathbf{c}_{bill}$ and $\mathbf{c}_{reward}$ encrypting the bills and rewards for each time slot.

$\mathbf{u} = (\mathsf{TP}_0, ..., \mathsf{TP}_{S-1}) \in \mathbb{Z}^S$ ▷ Trading price per time slot
$\mathbf{v} = (\mathsf{RP}_0, ..., \mathsf{RP}_{S-1}) \in \mathbb{Z}^S$ ▷ Retail price per time slot
$\mathbf{y} = (\mathsf{FiT}_0, ..., \mathsf{FiT}_{S-1}) \in \mathbb{Z}^S$ ▷ Feed-in tariff per time slot
$\mathbf{w} = \sum_{u=1}^{k} \mathbf{md}_u \in \mathbb{Z}^S$ ▷ Total deviation per time slot
Let $\mathbf{w}^{(0)} \in \{0,1\}^S$ such that $\mathbf{w}^{(0)}[i] = 1$ iff $\mathbf{w}[i] = 0$
Let $\mathbf{w}^{(<0)} \in \{0,1\}^S$ such that $\mathbf{w}^{(<0)}[i] = 1$ iff $\mathbf{w}[i] < 0$
Let $\mathbf{w}^{(>0)} \in \{0,1\}^S$ such that $\mathbf{w}^{(>0)}[i] = 1$ iff $\mathbf{w}[i] > 0$
$\mathbf{c}'_b = \mathsf{HE.NOT}(\mathbf{c}_b)$
▷ Block corresponding to $\mathsf{TD}_i = 0$
$\mathbf{c}^{(0)}_{bill} = \mathbf{c}_c \cdot \mathbf{u}$
$\mathbf{c}^{(0)}_{rew} = \mathbf{c}_s \cdot \mathbf{u}$
▷ Block corresponding to $\mathsf{TD}_i < 0$
$\mathbf{c}^{(<0)}_{rew} = \mathbf{c}_s \cdot \mathbf{u}$
▷ If $\mathsf{InDev}_i \leq 0$
$\mathbf{c}^{(<0)}_{bill} = \mathbf{c}_b \cdot \mathbf{c}_c \cdot \mathbf{u}$
▷ If $\mathsf{InDev}_i > 0$
$\mathbf{c}^{(<0)}_{bill} \mathrel{+}= \mathbf{c}'_b \cdot (\mathbf{c}_c \cdot \mathbf{u} + \frac{\mathbf{w}}{\mathsf{P2P}^c_n} \cdot (\mathbf{v} - \mathbf{u}))$
▷ Block corresponding to $\mathsf{TD}_i > 0$
$\mathbf{c}^{(>0)}_{bill} = \mathbf{c}_c \cdot \mathbf{u}$
▷ If $\mathsf{InDev}_i \leq 0$
$\mathbf{c}^{(>0)}_{rew} = \mathbf{c}_b \cdot \mathbf{c}_s \cdot \mathbf{u}$
▷ If $\mathsf{InDev}_i > 0$
$\mathbf{c}^{(>0)}_{rew} \mathrel{+}= \mathbf{c}'_b \cdot (\mathbf{c}_s \cdot \mathbf{u} + \frac{\mathbf{w}}{\mathsf{P2P}^c_n} \odot (\mathbf{y} - \mathbf{u}))$
▷ Combine bills and rewards for all possible TD_i
$\mathbf{c}_{bill} = \mathbf{c}^{(0)}_{bill} \cdot \mathbf{w}^{(0)} + \mathbf{c}^{(<0)}_{bill} \cdot \mathbf{w}^{(<0)} + \mathbf{c}^{(>0)}_{bill} \cdot \mathbf{w}^{(>0)}$
$\mathbf{c}_{reward} = \mathbf{c}^{(0)}_{rew} \cdot \mathbf{w}^{(0)} + \mathbf{c}^{(<0)}_{rew} \cdot \mathbf{w}^{(<0)} + \mathbf{c}^{(>0)}_{rew} \cdot \mathbf{w}^{(>0)}$

Publication 6

Reusable, Instant and Private Payment Guarantees for Cryptocurrencies

Publication Data

Akash Madhusudan, Mahdi Sedaghat, Samarth Tiwari, Kelong Cong and Bart Preneel, "Reusable, Instant and Private Payment Guarantees for Cryptocurrencies" In Information Security and Privacy - 2023 Australasian Conference, ACISP 2023, Lecture Notes in Computer Science, Springer-Verlag, pp. 580-605, 2023.

Contributions

- I was the main author together with Mahdi Sedaghat involved in Conceptualization, Methodology, Writing - Original Draft, Software, and Visualization.

Reusable, Instant and Private Payment Guarantees for Cryptocurrencies

Akash Madhusudan, Mahdi Sedaghat, Samarth Tiwari, Kelong Cong, and Bart Preneel

KU Leuven, ESAT/COSIC and imec, Leuven, Belgium and CWI, Amsterdam
{akash.madhusudan, ssedagha, kcong, bart.preneel}@esat.kuleuven.be,
{samarth.tiwari}@cwi.nl

Abstract. Despite offering numerous advantages, public decentralized cryptocurrencies such as Bitcoin suffer from scalability issues such as high transaction latency and low throughput. The vast array of so-called Layer-2 solutions tackling the scalability problem focus on throughput, and consider latency as a secondary objective. However, in the context of retail payments, instant finality of transactions is arguably a more pressing concern, besides the overarching concern for privacy.

In this paper, we provide an overlay network that allows privacy-friendly low latency payments in a retail market. Our approach follows that of a recent work called Snappy, which achieved low latency but exposed identities of customers and their transaction histories. Our construction ensures this data is kept private, while providing merchants with protection against double-spending attacks. Although our system is still based upon customers registering with a collateral, crucially this collateral is reusable over time.

The technical novelty of our work comes from randomness-reusable threshold encryption (RRTE), a cryptographic primitive we designed specifically for the following features: our construction provably guarantees payments to merchants, preserves the secret identity of honest customers and prevents their transactions from being linked. We also present an implementation of our construction, showing its capacity for fast global payments in a retail setting with a delay of less than 1 second.

1 Introduction

Public decentralized cryptocurrencies such as Bitcoin and Ethereum offer increased transparency and avoid trust in a central party. However, these advantages come at the

cost of performance, rendering them unfit for high throughput, real-time applications. The throughput of cryptocurrencies is orders of magnitude lower than that of traditional payment service providers such as Visa. Further, transactions require time before being considered final: the convention has been to wait for 6 confirmations or approximately an hour. For this reason, securely improving throughput and transaction confirmation latency (hereon referred to as latency) of public cryptocurrencies is a major area of research.

Layer-2 solutions tackle these constraints by offloading some elements of transactions off-chain [27]. However, these solutions focus on maximizing throughput and not on minimizing latency, which is dealt with as a secondary objective. Rollups, an increasingly popular solution in both industry and academia alike, increase the transaction throughput of Ethereum up to 100 times compared to its current throughput [12, 22, 8]. Yet, their latency matches that of the underlying blockchain. Payment Channel Networks (PCNs), another popular scaling solution, allow for arbitrarily many transactions on their network at the cost of a constant number of on-chain transactions, thereby drastically increasing throughput and reducing transaction fees. Yet, there are various factors negatively impacting the latency of PCN transactions, such as liveness of intermediate nodes [20], and the route discovery mechanisms [40]. Retail payments is a scenario where instant finality (hereon referred to as fast payments) is of the utmost importance; waiting an hour for a coffee is not an option. Additionally, retail payments are usually unilateral, i.e., from customers to merchants. An acknowledged problem with PCNs is channel depletion i.e., repeated use of channels in the same direction results in depleted channels, prohibiting further payments in the same direction [4]. Unilateral retail transactions only aggravate this issue. Similarly, by updating its layer-1, Ethereum 2.0 has increased its transaction throughput significantly. However, latency still takes $\sim$ 14 minutes [23]. This suggests a decoupling of two performance measures, namely throughput and latency, which need to be tackled separately.

Hence, *collateral reusability* and *fast payments* become interesting properties for solutions that perform retail payments with cryptocurrencies. Snappy is a fast on-chain payment system designed for a retail environment, where payers can assure payees of their ability to pay for a certain commodity [35]. Payees are protected from double-spending at rates much faster than the underlying blockchain while allowing collaterals to be reused. In order to work, Snappy depends on a set of *statekeepers* responsible for proactively detecting double-spending attempts in the system. These statekeepers have access to all transactions made by the customer, either by simply querying the blockchain or locally logging them when they receive it. Thus, their system suffers from privacy issues and all transactions involving the same payer can be linked together by third parties, such as, the statekeepers. The general demand for greater privacy becomes even more relevant in the retail context. For instance, the retail giant Target was able to deduce the pregnancy of a teenager even before her own

parents found out [31]. Such unfortunate leaks can be prevented, if merchants are unable to link customers' various purchases. Hence, there is a need for a solution that has the following properties:

P1. Instant payments with double-spending protection that is much faster than the underlying blockchain (*fast payments*).

P2. Prevents third parties in the system from being able to link different transactions involving the same honest payer (*transaction unlinkability*).

P3. Privacy for honest customers is provided efficiently without incurring latency constraints of payments (*efficient privacy*).

P4. Honest payers do not need to replenish their collaterals (*reusability*).

At ICDCS'21, Ng et al. [38] proposed LDSP that adds privacy to Snappy. LDSP achieves **P1**, **P2** and **P3**; however, at the cost of reusable collaterals (**P4**). Thus, previous works either trade-off privacy in order to achieve fast payments with collateral reusability or vice-versa.

Our Contributions. In this paper, we provide a new overlay network that fulfills all aforementioned properties. To the best of our knowledge, we are the first to simultaneously achieve the combination of properties (**P1-4**) in a payment system utilised in a retail context. Our contributions may be split into three as follows:

Private Low Latency Double-Spending Protection. We provide an overlay network capable of providing payees protection from double-spent transactions much faster than the underlying blockchain. No party in the system is able to link different transactions involving the same honest payer, nor do these privacy guarantees adversely affect latency of transactions. Customer collaterals are also *reusable*, so that customers can repeatedly guarantee payments over time.

Formal Security Analysis. We formally prove that our construction preserves the secret identity of honest customers (anonymity) and disables anyone from linking their transactions (unlinkability), yet at the same time guarantees payment to the merchants (payment certainty).

Implementation and evaluation. We implement our construction and show that it allows for fast global retail payments with a delay of less than 1 second.

Outline. The rest of this paper is organized as follows: Section 2 gives an overview of our construction, its participants and how they interact. Section 3 formally describes our construction, its threat model and proves how our construction satisfies its security requirements. Section 4 depicts an efficient instantiation of our construction and lists the cryptographic primitives used in detail. In Section 5 we evaluate the performance of

this instantiation while focusing on latency. In Section 6 we discuss our limitations and point out the differences our construction has when compared to similar state-of-the-art research and finally in Section 7 we conclude our paper.

2 Our construction

2.1 Design Motivation

Snappy

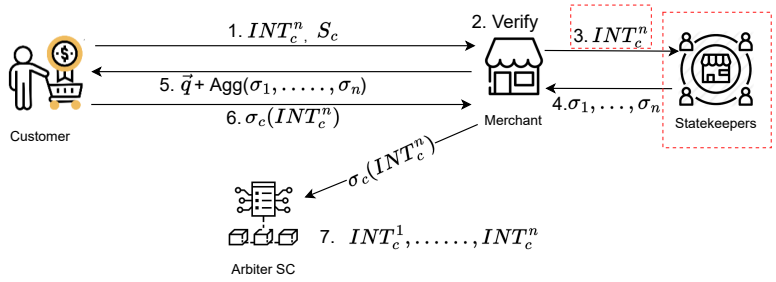
In this section, we only give a high-level description of a transaction in Snappy in order to point out its privacy shortcomings. For a detailed understanding of their construction, interested readers are referred to the Snappy paper [35].

In Snappy, a customer initiates a payment intent (INT_c) for a merchant of their choice. INT_c can be treated as a *payment guarantee* and is a normal blockchain transaction which is not yet signed by the customer. Hence, it includes typical transaction details such as the pseudonyms of payer and payee, transaction amount, etc. Snappy considers payment privacy to be orthogonal to their work, and claim that it is inherited from the underlying blockchain; however, by sharing INT_c with the statekeepers, payment privacy is also not provided on a protocol-level. In fig. 6.1a, all interactions impacting protocol-level privacy have been highlighted with a red dotted rectangle. By protocol-level privacy, we mean the privacy for payer and payee interactions from all third parties within the protocol. Even honest-but-curious statekeepers could potentially track all payments a customer makes without the need of eavesdropping, as this data readily becomes available to them.

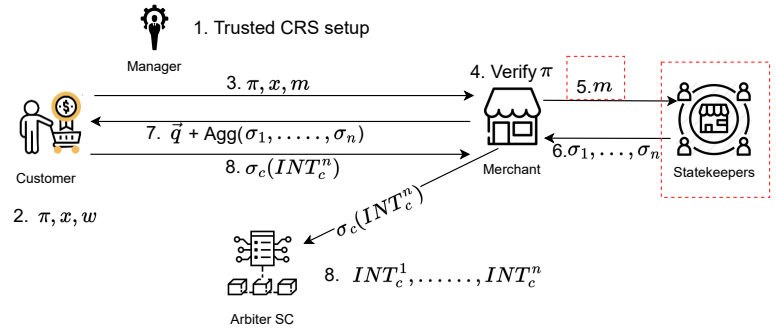
2.2 Strawman Construction

Here, let us see why a naive application of ZKPs to Snappy does not fulfil our desired properties.

As envisioned by Mavroudis et al. [35], one way to provide on-chain privacy in Snappy is to utilize privacy-preserving blockchains combined with private smart contracts. However, as we mention before, in addition to on-chain privacy, Snappy does not offer any protocol-level privacy as well. Keeping on-chain privacy orthogonal, an obvious solution to provide protocol-level privacy in Snappy would be to replace all sensitive and public information in the transactions on Snappy with zk-SNARKs. Note that, due to the choice of signature scheme in Snappy, i.e., ECDSA [32] and BLS [9], zk-SNARKs are the only option in terms of ZK proofs since less expressive proof



(a) Snappy overview.



(b) Snappy + zk-SNARKS.

Figure 6.1: Snappy's construction and a naive attempt to provide protocol-level privacy

systems, such as, Groth-Sahai proofs cannot work in combination with non-algebraic constraints, such as, hash functions.

Such a construction is shown in fig. 6.1b, which works as follows:

1. A trusted manager performs the one-time trusted setup required to use zk-SNARKs. Note that although alternatives to avoid this trusted setup exist [11, 34], their performance is not practical for use in real-time applications.

2. The customer generates a proof π of sufficient funds to pay the merchant, in addition to a statement x and witness $\hat{w}$.

3. The customer keeps $\hat{w}$ private, and forwards π and x to the merchant, along with a transaction counter m for their new payment intention. Note that m is similar to the index of INT_c in the original Snappy construction.

4. Upon receiving π, x and m, the merchant verifies π and is ensured that the customer has sufficient funds to pay for the services/product.

5. The merchant forwards m to the statekeepers in order for them to check whether they have seen it before or not (proactive double-spending protection).

The rest of the protocol proceeds similar to the original snappy protocol.

Observe that m is an index of the payment intention INT_c; hence, the payments from two different customers can have an identical m. As a result, statekeepers cannot verify whether or not a payment intention is a double-spend without also having access to the customer's identity.

Despite utilizing zk-SNARKs, the customer's identity *cannot* be hidden from statekeepers in Snappy, whilst using their current approach for proactive double-spending. Clearly, there is a need for a new way of (proactively) tracking double-spending attempts in order to preserve protocol-level privacy (in combination with ZKP). Additionally, assuming this new technique was to be included in Snappy's construction, there would still be a big constraint of inefficient ZK proof generation for customers. Since, the time required to generate a proof using zk-SNARKs is linear in the order of circuit size, the customer would spend substantial time in order to create a zk-SNARK for each INT_c.

2.3 Our solution

The strawman solution shown in Section 2.2 depicts the difficulty of a straightforward solution in achieving **P1-4**.

While designing a solution that fulfills all aforementioned properties, we faced two challenges. First, our construction must utilize a proactive double-spending protection mechanism that is able to *selectively* reveal information of dishonest parties while still keeping all information about honest parties private. Second, since the payment latency needs to be low, we must enable payers to prove vital information privately yet very efficiently. We address the first challenge by proposing a novel variant of threshold encryptions, which we hereon refer to as randomness-reusable threshold encryptions (RRTE) that has the unique property of revealing the plaintext if two ciphertexts are encrypted using the same randomness. We use RRTE in combination with Pseudo-Random Functions (PRF) such that the statekeepers are able to proactively protect from double-spend attempts without having knowledge of any transaction details; yet, are still able to reveal vital information in the case of double-spent transactions. The second challenge is addressed by a combination of Threshold Structure-Preserving Signatures (TSPS) [17] and Non-Interactive Zero-Knowledge Proofs (NIZKs). The compatibility of TSPS with efficient NIZKs enables a payer to prove vital information to payees in our system in zero-knowledge, without impacting the latency of transactions.

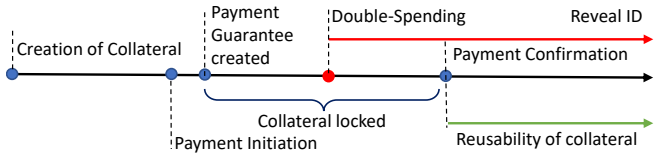


Figure 6.2: Timeline of a transaction

Participants. First, in order to explain the interplay of these cryptographic primitives that solve the aforementioned challenges and fulfill **P1-4**, we introduce the participants of our construction. Similar to Snappy, our participants include: (1) customers willing to purchase products/services using their cryptocurrencies while expecting low latency; (2) an established consortium of merchants willing to accept cryptocurrency payments and (3) statekeepers who are selected from the merchant consortium. Additionally, we also include a group of authorities *trusted* for registration of users and verification of collaterals. Although authorities have greater power during the setup of our network, they *do not* have any access to future transactions between a customer and a merchant. For more discussion on these trust assumptions see Section 6.

Interaction of Participants. In our construction, the participants function as follows; each *customer* in our systems owns collateral(s), which is uniquely linked to a secret PRF key(s). This secret PRF key is signed by the group of *authorities* by utilizing TSPS once they verify its existence on the arbiter smart contract. The PRF is used in combination with our RRTE to set up our double-spending protection. By using this signed secret PRF key, each customer generates a randomness, which they then use in combination with the public key of *target merchant* to encrypt their secret identity. For encryption we use RRTE, that reveals the secret identity of a malicious customer (plaintext) when they double-spend. More precisely, double-spending in our system amounts to certifying multiple transactions with the same collateral, by which RRTE reveals the dishonest customer's identity. Thus, our construction fulfills **P1** by enabling any *statekeeper* who receives two ciphertexts that are encrypted using the same randomness to catch double-spending and reveal the identity of the perpetrator. However, honest customers who only certify one transaction per collateral remain unaffected; hence, also fulfilling **P2**.

Next, the customer needs to convince the target merchant about the existence of their collateral. However, they must do this without revealing any identifying details. To achieve this, we utilize TSPS and NIZKs. The payment guarantee is an indirect proof of collateral by proving knowledge of the TSPS provided to them by the authorities. In case of a double-spend attempt, this payment guarantee is sufficient for the merchant to be reimbursed. This can be done efficiently by efficient proof systems fulfilling **P3**.

Finally, our construction also allows a customer to *reuse* their collaterals after a certain

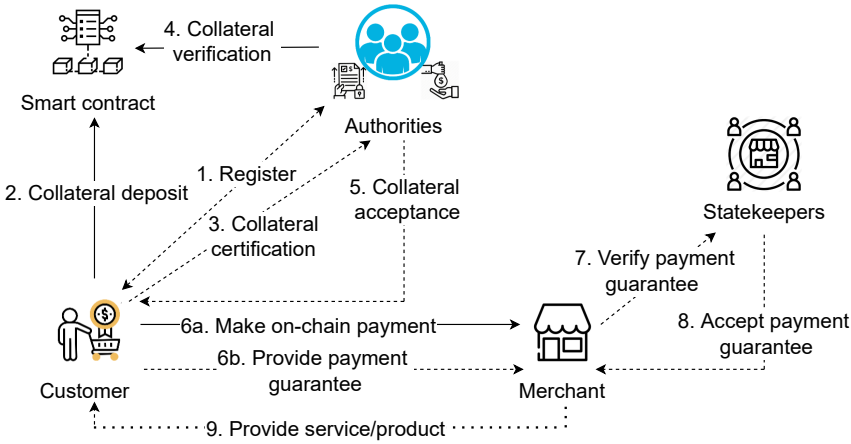


Figure 6.3: Our construction

time interval in order to achieve **P4**. As illustrated in fig. 6.2, a collateral is considered *locked* for the time period between the customer sending a payment guarantee to the merchant and the actual confirmation of their blockchain transaction. However, after the confirmation of a blockchain transaction, the customer's collateral can be reused. Note that this is similar to the execution of a multi-hop payment on PCNs, that rely on Hashed-Time-Locked-Contracts, locking up the collateral for a similar time period [40, 2]. Thus, using a collateral twice in quick succession is treated as a double-spend and can be detected, but it may be reused once the locking period of the collateral has elapsed.

Construction Overview. fig. 6.3 illustrates our construction. In order to explain our construction, we assume an established set of authorities and a fixed merchant consortium. The payment protocol between a customer and a merchant has off-chain and on-chain components that have been depicted in fig. 6.3 with dotted and solid arrows respectively. The protocol proceeds as follows:

1) The customer begins by registering themselves with a set of authorities. **2)** The customer deposits a collateral in the smart contract, controlled by this set of authorities. **3)** Once the deposit is confirmed on the blockchain, the customer requests a collateral certification, which is necessary to generate payment guarantees for merchants. **4)** Upon receiving the certification request of the customer, the authorities check the smart contract to confirm if the customer deposited a collateral. **5)** The authorities provide the customer with a signed collateral certification. **6a)** During a transaction in the retail market, the customer makes a payment to the target merchant using the underlying cryptocurrency. This is done by sending a payment to the merchant through the smart contract. **6b)** The customer simultaneously generates an encrypted *payment guarantee*

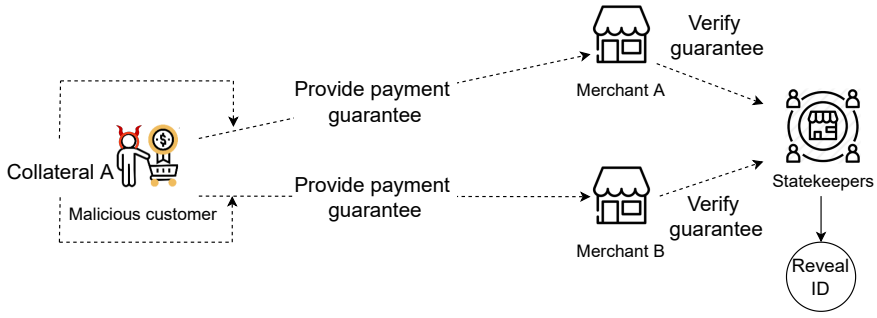


Figure 6.4: Proactive double-spending detection

by using the collateral certification and sends this to the target merchant along with the transaction identifier of their cryptocurrency payment. **7)** The target merchant forwards this encrypted *payment guarantee* to the statekeepers who individually confirm that it is not a double-spend attempt. The statekeepers compare the received guarantee with each guarantee they received within a fixed time period. If none of these comparisons reveal the plaintext, i.e, secret identity of the customer, they are guaranteed about its uniqueness. Note that this verification can be done by utilizing our proposed RRTE and does not require knowing a customer's identity. **8)** Each statekeeper returns a signed payment guarantee to the merchant. **9)** If a majority of the statekeepers return a confirmation, the merchant aggregates these signatures and accepts the payment guarantee and provides the customer with necessary services/products.

Double-Spending. Double-spending in our construction means a scenario where the customer is malicious and wants to double-spend their payment guarantee, i.e., use the same collateral to promise payments to two distinct merchants. In this case, when the statekeepers receive these guarantees from two distinct merchants, they are able to combine them and reveal the secret identity of the cheating customer. Along with this identity, a secret to the customer's collateral is also revealed. This secret is used by the victim merchant for remuneration. This is depicted in fig. 6.4

More details about how RRTE enables statekeepers to reveal the secret identity of a malicious customer are given in its formal definition in section 4.

Limitations. Although we address the privacy shortcomings of Snappy's design, our construction still inherits some of its other limitations, such as, the large collateral requirement for statekeepers and inadequately defined user incentives. To be specific, we also require the merchants to deposit collaterals in order to play the role of statekeepers fairly. We also assume that the amount of a customer's deposit is *fixed*. The assumption of a fixed set of merchants is also an additional concern, since it does not allow for any merchant churn in the protocol. However, in this paper we do not attempt

to address these concerns. A more thorough discussion about these shortcomings is given in Section 6.

3 Preliminaries and Formal Construction

Throughout this paper, we let the security parameter of the scheme to be λ with unary representation of 1^λ, and $\mathsf{negl}(\lambda)$ denotes a negligible function. We use $x \leftarrow\!\$\, X$ to denote that x is sampled uniformly from the set X. $[n]$ denotes the set of integers in the range of 1 to n. For clarity, the secret values in our construction are represented with a *hat* operator (e.g., $\hat{\mathsf{sk}}$) and masked values are represented with the notation x' for the value x.

Threat Model. Our construction is designed to resist active adversaries that can corrupt a set of customers, merchants (which are also statekeepers) and authorities. To be precise, an adversary can only corrupt a minority of authorities during the initialization phase. During the processing of transactions, the adversary can corrupt all but one customers, and a minority of merchants.

Network Assumptions. We assume the existence of secure and reliable communication channels so that parties receive messages sent by honest parties eventually. The honest merchants/statekeepers are also assumed to be live and responsive, for the verification of transactions. We do not expect the customers to be online, unless they need to transact with a merchant. The underlying blockchain is assumed to be persistent and live and the adversary cannot influence the consensus mechanism.

Beyond the threat model. Let us mention a few considerations that are outside the scope of our model. Firstly, we consider protocol-level privacy, and not the privacy of the underlying blockchain. We do not consider side-channel attacks, such as metadata-level attacks on communication channels. Finally, the selection of authorities is also orthogonal to our work. One possibility is to choose them from the set of reputable merchants in the consortium, and have them shuffled periodically to ensure a majority of authorities always remain honest.

We would also like to note that weaker assumptions of trust or liveness are possible without compromising security. However, we select a simple set of assumptions, following those of Snappy, in order to focus on the privacy aspect instead of system-level optimizations.

Formal Construction. Our construction builds on Pseudo-Random Function (PRF), Non-Interactive Zero-Knowledge (NIZK) arguments, Digital Signatures (DS), Threshold Structure-Preserving Signatures (TSPS), Commitments (CO), and a novel randomness-reusable Threshold Encryption (RRTE). We list the formal definition and the security properties in appendix A.1 and outline the scheme in fig. 6.5 for a

relation $\mathbf{R_L}$ over three main NP-languages $\mathbf{L} := (\mathbf{L}_1, \mathbf{L}_2, \mathbf{L}_3)$. The list of master public keys, mpk, is considered as an implicit input for all algorithms except the parameter generation algorithm (PGen). Additionally, all algorithms are PPT unless otherwise specified. We now formalise the functions in the fig. 6.5. All functions are split based on when they happen in our construction and are formalized as follows:

Bootstrapping Phase as depicted in fig. 6.3:

- $\mathsf{PGen}(1^\lambda, \mathbf{R_L})$ takes λ in its unary representation and relation $\mathbf{R_L}$ as inputs and returns the master public key mpk.
- $\mathsf{AuKeyGen}(\mathcal{AU})$ is executed by $\mathcal{AU}$ that returns $(\hat{\mathsf{sgk}}_{ai}, \mathsf{vk}_{ai})$ for $i \in [n]$ along with a global verification key vk_a. $\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}, \mathsf{vk}_{ai}, \mathsf{vk}_a]_{i=1}^{n}$ represents the list of credentials for $\mathcal{AU}$.
- $\mathsf{MKeyGen}(\mathsf{M}_m)$ is executed by merchant $\mathsf{M}_m \in \mathcal{M}$ in order to join the network. It initially generates a pair of signing/verification keys $(\hat{\mathsf{sgk}}_{bm}, \mathsf{vk}_{bm})$ and returns a tuple $(\hat{\mathsf{sgk}}_{bm}, \mathsf{vk}_{bm}, \mathsf{pk}_{bm} = \perp)$. The list of keys belonging to the group of merchants is recorded in $\mathcal{M}[\mathsf{sgk}_{bi}, \mathsf{vk}_{bi}]_{i=1}^{\ell}$.
- $\mathsf{MRegister}(\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}]_{i=1}^{t}, \mathcal{M}[\mathsf{vk}_{bi}]_{i=1}^{\ell})$ is executed by any subset of $\mathcal{AU}$ of size at least t to register the merchants who deposit a collateral to join the merchant consortium and assign them a public key pk_{bm}. For each merchant $\mathsf{M}_m \in \mathcal{M}$, it takes the secret signing key of the authorities $\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}]$ for $1 \leq i \leq t$, and returns public key pk_{bm} as output. After this phase, the list of parameters for the mth merchant can be updated as $\mathcal{M}[\hat{\mathsf{sgk}}_{bm}, \mathsf{vk}_{bm}, \mathsf{pk}_{bm}]$.
- $\mathsf{CuKeyGen}(\mathsf{C}_n)$ is executed by the customers, that for each customer $\mathsf{C}_n \in \mathcal{C}$, a Pseudo-ID generator function (PID) generates an initial secret key $\hat{\mathsf{sk}}_{cn}$ and its corresponding public key pk_{cn}. It returns a tuple of $(\mathsf{pk}_{cn}, \hat{\mathsf{sk}}_{cn}, \hat{\mathsf{cert}}_{cn} = \perp)$ with a NIZK proof π_1 to prove that the relation $\mathbf{R}_{\mathbf{L}_1}$ fulfills. The list of customers' keys are kept in $\mathcal{C}[\hat{\mathsf{sk}}_{ci}, \mathsf{pk}_{ci}, \hat{\mathsf{cert}}_{ci} = \perp]_{i=1}^{k}$.

Protocol as depicted in fig. 6.3:

- $\mathsf{CuRegister}(\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}]_{i=1}^{t}, \mathcal{C}[\mathsf{pk}_{cn}], \pi_1)$ is depicted in step 1 of fig. 6.3. It is executed by any subset of $\mathcal{AU}$ of size at least t to certify the public key pk_{cn} of a customer C_n corresponds to some secret value $\hat{\mathsf{sk}}_{cn}$ under the relation $\mathbf{R}_{\mathbf{L}_1}$. Once the customer C_n is registered by the authorities, it receives a certificate $\hat{\mathsf{cert}}_{cn}$ and it updates $\mathcal{C}[\mathsf{pk}_{cn}, \hat{\mathsf{sk}}_{cn}, \hat{\mathsf{cert}}_{cn}]$.
- $\mathsf{CuCreate}(\mathcal{C}[\hat{\mathsf{sk}}_{cn}, \hat{\mathsf{cert}}_{cn}])$ is depicted in step 2-3 of fig. 6.3. It is executed by the customer to request for certification of their collateral. A successfully registered

Algorithm 1: OUR CONSTRUCTION.

Function $\texttt{PGen}(1^\lambda, \mathbf{R_L})$:
- $(\vec{\mathsf{crs}}, \vec{\mathsf{ts}}, \vec{\mathsf{te}}) \leftarrow \mathcal{ZK}.\mathsf{K}_{\vec{\mathsf{crs}}}(1^\lambda, \mathbf{R_L})$
- $(\mathsf{pp}) \leftarrow \mathcal{TSPS}.\mathsf{Setup}(1^\lambda)$
- **return** $\mathsf{mpk} := (\mathsf{pp}, \vec{\mathsf{crs}})$

Function $\texttt{AuKeyGen}(\mathcal{AU})$:
- $(\hat{\vec{\mathsf{sgk}}}_a, \vec{\mathsf{vk}}_a, \mathsf{vk}_a) \leftarrow \mathcal{TSPS}.\mathsf{KGen}(\mathsf{mpk}, n, t)$
- **return** $(\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}, \mathsf{vk}_{ai}, \mathsf{vk}_a]_{i=1}^n)$

Function $\texttt{MKeyGen}(\mathsf{M}_m)$:
- $(\hat{\mathsf{sgk}}_{bm}, \mathsf{vk}_{bm}) \leftarrow \mathcal{DS}.\mathsf{KGen}(\mathsf{pp})$
- **return** $(\mathcal{M}[\hat{\mathsf{sgk}}_{bm}, \mathsf{vk}_{bm}])$

Function $\texttt{MRegister}(\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}]_{i=1}^t, \mathcal{M}[\mathsf{vk}_{bi}]_{i=1}^\ell)$:
- **for** $j \in \mathrm{range}(\ell)$ **do**
  - $(\mathsf{pk}_{bj}, \mathsf{pk}_b) \leftarrow \mathcal{RRTE}.\mathsf{KGen}(\mathsf{mpk}, \ell, t, 2)$
- **return** $(\mathcal{M}[\mathsf{pk}_{bi}]_{i=1}^\ell, \mathsf{pk}_b)$

Function $\texttt{CuKeyGen}(\mathsf{C}_n)$:
- $\hat{\mathsf{sk}}_{cn} := \mathsf{PID}(\mathsf{C}_n) \in \mathbb{Z}_\mathsf{p}^*$
- $\mathsf{sk}'_{cn} \leftarrow \mathcal{CO}.\mathsf{Com}(\mathsf{pp}, \hat{\mathsf{sk}}_{cn})$
- $\mathsf{pk}_{cn} := (\mathsf{sk}'_{cn}, M_1, M_2) \leftarrow \mathsf{iDH}^{\mathsf{H}}(\mathsf{sk}'_{cn}, \hat{\mathsf{sk}}_{cn})$
- $\mathsf{x}_1 = (\mathsf{pk}_{cn})$
- $\hat{\mathsf{w}}_1 = (\hat{\mathsf{sk}}_{cn})$
- $\pi_1 \leftarrow \mathcal{ZK}.\mathsf{P}(\mathbf{R}_{\mathbf{L}_1}, \vec{\mathsf{crs}}, \mathsf{x}_1, \hat{\mathsf{w}}_1)$
- **return** $(\mathcal{C}[\mathsf{pk}_{cn}, \hat{\mathsf{sk}}_{cn}], \pi_1)$

Function $\texttt{CuRegister}(\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}]_{i=1}^t, \mathcal{C}[\mathsf{pk}_{cn}], \pi_1)$:
- **if** $\mathcal{ZK}.\mathsf{Vf}(\mathbf{R}_{\mathbf{L}_1}, \vec{\mathsf{crs}}, \mathsf{x}_1, \pi_1) = 1$ **then**
  - $(\hat{\mathsf{cert}}_{cn}) \leftarrow \mathcal{TSPS}.\mathsf{Sign}(\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}]_{i=1}^t, \mathsf{pk}_{cn})$
- **return** $(\mathcal{C}[\hat{\mathsf{cert}}_{cn}])$

Function $\texttt{CuCreate}(\mathcal{C}[\hat{\mathsf{sk}}_{cn}, \hat{\mathsf{cert}}_{cn}])$:
- $\hat{k}_j \leftarrow \mathcal{PRF}.\mathsf{KGen}(\mathsf{pp})$
- $k'_j \leftarrow \mathcal{CO}.\mathsf{Com}(\mathsf{pp}, \hat{k}_j)$
- $M_j := (k'_j, M_1, M_2) \leftarrow \mathsf{iDH}^{\mathsf{H}}(k'_j, k_j)$
- $\mathsf{x}_2 = (k'_j)$
- $\hat{\mathsf{w}}_2 = (\hat{k}_j, \hat{\mathsf{sk}}_{cn}, \hat{\mathsf{cert}}_{cn})$
- $\pi_2 \leftarrow \mathcal{ZK}.\mathsf{P}(\mathbf{R}_{\mathbf{L}_2}, \vec{\mathsf{crs}}, \mathsf{x}_2, \hat{\mathsf{w}}_2)$
- **return** $(\mathcal{CL}[\hat{k}_j, M_j], \mathcal{C}[\mathsf{cert}'_{cn}], \pi_2)$

Function $\texttt{AuCreate}(\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}, \mathsf{vk}_a]_{i=1}^t, \mathcal{C}[\mathsf{cert}'_{cn}], \mathcal{CL}[\hat{k}_j, k'_j], \pi_2)$:
- **if** $\mathcal{ZK}.\mathsf{Vf}(\mathbf{R}_{\mathbf{L}_2}, \vec{\mathsf{crs}}, \mathsf{x}_2, \pi_2) = 1$ **then**
  - $(\hat{\mathsf{cert}}_j) \leftarrow \mathcal{TSPS}.\mathsf{ParSign}(\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}]_{i=1}^t, M_j)$
- **return** $(\mathcal{CL}[\hat{\mathsf{cert}}_j])$

Function $\texttt{Spend}(\mathcal{C}[\hat{\mathsf{cert}}_{cn}, \hat{\mathsf{sk}}_{cn}], \mathcal{CL}[\hat{k}_j, \hat{\mathsf{cert}}_j], \mathcal{M}[\mathsf{pk}_{bm}], \mathsf{t})$:
- $(r_\mathsf{t}) \leftarrow \mathsf{PRF}_{\hat{k}_j}(\mathsf{t})$
- $R_\mathsf{t} := e(r_\mathsf{t}, h)$ ▷ h is the generator of $\mathbb{G}_2$.
- $\mathtt{Ct}_m \leftarrow \mathcal{RRTE}.\mathsf{Enc}(\mathsf{pk}_{bm}, \hat{\mathsf{sk}}_{cn}; r_\mathsf{t})$
- $\hat{\mathsf{w}}_3 = (\hat{\mathsf{cert}}_j, \hat{k}_j, r_\mathsf{t}, \hat{\mathsf{sk}}_{cn})$
- $\mathsf{x}_3 = (R_\mathsf{t}, \mathtt{Ct}_m, \mathsf{t})$
- $\pi_3 \leftarrow \mathcal{ZK}.\mathsf{P}(\mathbf{R}_{\mathbf{L}_3}, \vec{\mathsf{crs}}, \mathsf{x}_3, \hat{\mathsf{w}}_3)$
- **return** $(\mathcal{T}[\pi_3, \mathsf{x}_3, Tx_{ID}])$

Function $\texttt{Vf}(\mathcal{M}[\mathsf{pk}_{bm}], \mathcal{T}[\pi_3, \mathsf{x}_3, Tx_{ID}], \mathsf{t})$:
- **if** $\mathcal{ZK}.\mathsf{Vf}(\mathbf{R}_{\mathbf{L}_3}, \vec{\mathsf{crs}}, \mathsf{x}_3, \pi_3) = 1$ **then**
  - **for** $i \in \mathsf{StK}$ **do**
    - **if** $R_\mathsf{t} \notin \mathcal{L}_i$ **then**
      - $(\sigma_{R_\mathsf{t},i}) \leftarrow \mathcal{DS}.\mathsf{Sign}(\hat{\mathsf{sgk}}_{bi}, R_\mathsf{t})$
  - **if** $\mathcal{DS}.\mathsf{Vf}(\mathsf{vk}_{bi}, \sigma_{R_\mathsf{t},i}) = 1 \ \wedge \ |\sigma_{R_\mathsf{t}}| \geq (|\mathsf{StK}|/2) + 1$ **then**
    - **return** 1

Function $\texttt{RevealID}(\mathtt{Ct}_m, \mathtt{Ct}_{m'}, v)$:
- $(\hat{\mathsf{sk}}_{cn}) \leftarrow \mathcal{RRTE}.\mathsf{Dec}(\mathtt{Ct}_m, \mathtt{Ct}_{m'}, v)$
- **return** (sk_{cn})

Figure 6.5: Our Construction

customer C_n, with certificate $\hat{\mathsf{cert}}_{cn} \neq \perp$, can deposit collaterals in the smart contract. For each deposit j in the smart contract, the customer samples a random value k_j from a uniform distribution $\mathcal{K}_{\mathsf{PRF}}$ in a way that the deposit is not directly linkable to the customer. Then it returns a tuple $\mathcal{CL}[\hat{k}_j, k'_j, \perp]$ as an uncertified collateral along with a proof π_2 depicting the fact that the relation $\mathbf{R}_{\mathbf{L}_2}$ fulfills.

- $\mathsf{AuCreate}(\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}]_{i=1}^{t}, \mathcal{C}[\mathsf{cert}'_{cn}], \mathcal{CL}[\hat{k}_j, k'_j], \pi_2)$ is depicted in step 4-5 of fig. 6.3. It is executed by a group of authorities $\mathcal{AU}$ of size at least t. It takes the authorities' secret signing keys ($\hat{\mathsf{sgk}}_{ai}$), an indexed DH message space of PRF key and a NIZK proof π_2 as inputs. To create a certified collateral, it checks the validity of the proof π_2 and whether this collateral exists in the smart contract, and returns certificate $\hat{\mathsf{cert}}_j$ as output. The list of parameters for each collateral is kept by $\mathcal{CL}[\hat{k}_j, k'_j, \hat{\mathsf{cert}}_j]$.

- $\mathsf{Spend}(\mathcal{C}[\hat{\mathsf{sk}}_{cn}, \hat{\mathsf{cert}}_{cn}], \mathcal{CL}[\hat{k}_j, \hat{\mathsf{cert}}_j], \mathcal{M}[\mathsf{pk}_{bm}], \mathsf{t})$ is depicted in step 6a and 6b of fig. 6.3. It is executed by a customer $\mathsf{C}_n \in \mathcal{C}$ who performs a payment to the merchant $\mathsf{M}_m \in \mathcal{M}$ using the underlying cryptocurrency of their choice at time t. The registered customer uses a certified collateral $\mathcal{CL}[\hat{k}_j, \hat{\mathsf{cert}}_j]$ to provide a payment guarantee to the merchant M_m. The payment made by the customer is always bounded by a publicly known collateral amount. It returns the transaction details as a list of parameters $\mathcal{T}[\mathsf{x}_3, \pi_3, Tx_{ID}]$, which contains a pair of instance and proof (x_3, π_3), along with a set of auxiliary data R_{t}. This function is executed in parallel with an on-chain payment. In particular, the customer must first sign and broadcast an on-chain transaction Tx, and then include its identifier (Tx_{ID}) in the payment guarantee of Spend. The Tx_{ID} could have different formats depending on the underlying blockchain.[1]

- $\mathsf{Vf}(\mathcal{M}[\mathsf{pk}_{bm}], \mathcal{T}[\pi_m, \mathsf{x}_m, Tx_{ID}], \mathsf{t})$ is depicted in step 7-9 of fig. 6.3. The merchant $\mathsf{M}_m \in \mathcal{M}$ executes it to check the validity of a received payment guarantee. Once the proof is verified successfully by merchant M_m along with the majority of statekeepers, StK, confirmation that they have not seen a similar payment guarantee in the current epoch (by providing their signatures), the merchant verifies their individual signatures and aggregates them. Once the aggregation is complete, and if Tx_{ID} specifies the merchant's address as the receiver of funds, the merchant provides the items/services to the customer without waiting for the transaction confirmation of the customer's original payment on the blockchain. If the proof verification fails, or the majority of statekeepers do not confirm the guarantee, or the on-chain transaction specifies the wrong receiver address, the merchant rejects the payment guarantee.

Double-spend detection as depicted in fig. 6.4:

[1] For a public blockchain such as Bitcoin or Ethereum, Tx_{ID} could be the hash of a transaction (H[Tx]); for an anonymous blockchain like Zcash, it could be the viewing key of the transaction that enables the merchant to check if he is the receiver of the shielded transaction [21]

- $\mathsf{RevealID}(\mathtt{Ct}_m, \mathtt{Ct}_{m'}, v)$ is a deterministic algorithm that takes two ciphertexts $\mathtt{Ct}_m$ and $\mathtt{Ct}_{m'}$ generated under the public key of two distinct merchants M_m and $\mathsf{M}_{m'}$ and returns the plaintext, i.e., the identity of the customer and the secret to their collateral to redeem it. This ID, sk_{cn}, is no longer hidden and can be used by $\mathcal{AU}$ to blacklist the cheating customer and its collateral(s) can be used to remunerate the victim merchant.

3.1 NIZK Languages

In the proposed generic construction in fig. 6.5 we rely on three languages for the NIZK systems, described below.

- Language $\mathbf{L}_1$: Used to prove the correct formation of the customers' public key pk_{cn}, based on the knowledge of secret key $\hat{\mathsf{sk}}_{cn}$. We depict this language formally below, which is used during $\mathsf{CuKeyGen}(\mathsf{C}_n)$.

$$\mathbf{L}_1 = \mathsf{NIZK}\{(\hat{\mathsf{sk}}_{cn}) \mid \mathsf{sk}'_{cn} := \mathcal{CO}.\mathsf{Com}(\hat{\mathsf{sk}}_{cn})\}$$

- Language $\mathbf{L}_2$: Used to prove eligibility to request a collateral by deriving certificate fulfillment. This language is used during $\mathsf{CuCreate}(\mathcal{C}[\hat{\mathsf{cert}}_{cn}])$.

$$\mathbf{L}_2 = \mathsf{NIZK}\{(\hat{k}_j, \hat{\mathsf{sk}}_{cn}, \hat{\mathsf{cert}}_{cn}) \mid k'_j := \mathcal{CO}.\mathsf{Com}(\hat{k}_j), \hat{k}_j \in \mathcal{K}_{\mathsf{PRF}}, \mathcal{TSPS}.\mathsf{Vf}(\mathsf{pk}_{cn}, \hat{\mathsf{cert}}_{cn}) = 1\}$$

- Language $\mathbf{L}_3$: Used to prove the possession of a valid collateral, correctness of PRF evaluation algorithm and RRTE's ciphertext. This language is used during $\mathsf{Spend}(\mathcal{C}[\hat{\mathsf{sk}}_{cn}, \hat{\mathsf{cert}}_{cn}], \mathcal{CL}[\hat{k}_j, \hat{\mathsf{cert}}_j], \mathcal{M}[\mathsf{pk}_{bm}], \mathsf{t})$.

$$\begin{aligned}\mathbf{L}_3 = &\mathsf{NIZK}\{(\hat{\mathsf{cert}}_j, \hat{k}_j, r_\mathsf{t}, \hat{\mathsf{sk}}_{cn}) \mid r_\mathsf{t} \leftarrow \mathsf{PRF}_{\hat{k}_j}(\mathsf{t}), R_t := e(r_t, h),\\ &\mathtt{Ct}_m := \mathcal{RRTE}.\mathsf{Enc}(\mathsf{pk}_{bm}, \hat{\mathsf{sk}}_{cn}; r_\mathsf{t}), \mathcal{TSPS}.\mathsf{Vf}(M_j, \hat{\mathsf{cert}}_j) = 1\}\end{aligned}$$

3.2 Security Analysis

Next we formally define the two main security requirements for our construction, namely (1) Anonymity of honest customers and Unlinkability of payment guarantees, and (2) Payment certainty for honest merchants. Note that the $\mathsf{AllGen}(.)$ algorithm (see fig. 6.6) generates all system setup parameters at once. In the described definitions, it is implicitly assumed that there exists a PPT adversary $\mathcal{A}$ who has access to the following oracles provided by the challenger $\mathcal{B}$:

- **Oracle** $O_{\mathsf{AuCorrupt}}(Au_i)$: By calling this oracle under the input Au_i, $\mathcal{A}$ can corrupt Au_i and receive its internal states. The set of corrupted authorities is denoted by $\mathcal{AU}'$ and we have $|\mathcal{AU}'| < t$.
- **Oracle** $O_{\mathsf{CuCorrupt}}(.)$: Adversary $\mathcal{A}$ can corrupt any customer $\mathsf{C}_n \in \mathcal{C}$ by querying this oracle, and receive its uncertified secret key $\hat{\mathsf{sk}}_{cn}$.
- **Oracle** $O_{\mathsf{ColCorrupt}}(.)$: $\mathcal{A}$ can corrupt at most q_D collaterals $CL_j \in \mathcal{CL}$ to receive their uncertified secret value $\hat{k}_j$. The list of corrupted collaterals is represented by $\mathcal{CL}'$.
- **Oracle** $O_{\mathsf{MCorrupt}}(.)$: Adversary $\mathcal{A}$ can corrupt a minority set of merchants (statekeepers) like $\mathsf{M}_m \in \mathcal{M}$ and receives its pair of public key pk_{bm} and secret signing key $\hat{\mathsf{sgk}}_{bm}$. The list of corrupted merchants is denoted by $\mathcal{M}'$ s.t. we have, $|\mathcal{M}'| < |\mathsf{stk}|/2$.
- **Oracle** $O_{\mathsf{Revoke}}(.)$: Adversary $\mathcal{A}$ can revoke at most q_R certified collaterals $CL_j \in \mathcal{CL}$ and redeem the deposited money.
- **Oracle** $O_{\mathsf{Spend}}(.)$: $\mathcal{A}$ can make at most q_S payment guarantees created by any arbitrary non-corrupted customer to any non-corrupted merchant.

Definition 1 (Payment Unlinkability and Anonymity)**.** *This construction preserves the anonymity of honest customers and provides unlinkability of payment guarantees, if no PPT adversary $\mathcal{A}$ by getting access to $O_{\mathsf{AuCorrupt}}, O_{\mathsf{CuCorrupt}}, O_{\mathsf{MCorrupt}}, O_{\mathsf{Spend}}$ oracles, O_{ANON} in short, and with advantage of $Adv^{\mathrm{ANON}}_{\mathcal{A}}(\lambda,\beta) = 2\left((\mathrm{EXP}^{\mathrm{ANON}}_{\mathcal{A}}(1^{\lambda},\beta) = 1) - 1/2\right)$, has a non-negligible chance of winning the experiment described in fig. 6.6, i.e. we have, $\left|Adv^{\mathrm{ANON}}_{\mathcal{A}}(\lambda,\beta=0) - Adv^{\mathrm{ANON}}_{\mathcal{A}}(\lambda,\beta=1)\right| \leq \mathsf{negl}(\lambda)$.*

Definition 2 (Payment Certainty)**.** *This construction provides payment certainty (PC) if no transaction τ is approved with a non-negligible advantage s.t. $q_S + q_R + \tau > q_D$. No PPT adversary $\mathcal{A}$ with access to $O_{\mathsf{AuCorrupt}}, O_{\mathsf{CuCorrupt}}, O_{\mathsf{ColCorrupt}}, O_{\mathsf{Revoke}}, O_{\mathsf{Spend}}$ oracles, O_{PC} in short, can win the experiment described in fig. 6.6 with a non-negligible advantage in λ and we can write, $Adv^{\mathrm{PC}}_{\mathcal{A}}(\lambda) := \Pr[\mathrm{EXP}^{\mathrm{PC}}_{\mathcal{A}}(1^{\lambda}) = 1] \leq \mathsf{negl}(\lambda)$.*

As a consequence of payment unlinkability and anonymity, no PPT adversary can expose any information about the transaction such as the identity of honest customers or be able to link it to any other transaction made by the customer. As a consequence of payment certainty, no entity, not even after colluding with a group of participants, can transfer and/or revoke more money than the amount deposited. Finally, any system satisfying both these definitions simultaneously reveals the identity of a malicious customer attempting to use one collateral to pay multiple merchants at the same time.

Theorem 1. *The proposed generic construction in fig. 6.5 satisfies the unlinkability and anonymity of payment guarantees as defined in definition 1.*

$\text{EXP}^{\text{ANON}}_{\mathcal{A}}(1^\lambda, \beta)$

1 : $(\mathsf{mpk}, \mathsf{H}, \mathcal{AU}[\hat{\mathsf{sgk}}_{ai}, \mathsf{vk}_{ai}, \mathsf{vk}_a]_{i=1}^n, \mathcal{M}[\hat{\mathsf{sgk}}_{bi}, \mathsf{vk}_{bi}, \mathsf{pk}_{bi}]_{i=1}^\ell) \leftarrow \mathsf{AllGen}(1^\lambda, \mathbf{R_L})$

2 : $(\mathsf{M}_m, \hat{\mathsf{sk}}_0^*, \hat{\mathsf{sk}}_1^*, \hat{k}_0^*, \hat{k}_1^*) \leftarrow \mathcal{A}^{O_{\text{ANON}}}(\mathsf{mpk})$

3 : $\beta \leftarrow\$ \{0,1\}$

4 : $(\pi_\beta, \mathsf{x}_\beta, R_{\mathsf{t},\beta}) \leftarrow \mathsf{Spend}\big(\mathcal{C}[\hat{\mathsf{cert}}_{c\beta}^*, \hat{\mathsf{sk}}_\beta^*], \mathcal{CL}[\hat{k}_\beta^*, \hat{\mathsf{cert}}_\beta^*], \mathcal{M}[\mathsf{pk}_{bm}], \mathsf{t}\big)$

5 : $\beta' \leftarrow\$ \mathcal{A}^{O_{\text{ANON}}}\left(\mathcal{T}[\pi_\beta, \mathsf{x}_\beta, R_{\mathsf{t},\beta}]\right)$

6 : **return** $\beta' == \beta$

$\text{EXP}^{\text{PC}}_{\mathcal{A}}(1^\lambda)$

1 : $(\mathsf{mpk}, \mathsf{H}, \mathcal{AU}[\hat{\mathsf{sgk}}_{ai}, \mathsf{vk}_{ai}, \mathsf{vk}_a]_{i=1}^n, \mathcal{M}[\hat{\mathsf{sgk}}_{bi}, \mathsf{vk}_{bi}, \mathsf{pk}_{bi}]_{i=1}^\ell) \leftarrow \mathsf{AllGen}(1^\lambda, \mathbf{R_L})$

2 : $(\pi^*, \mathsf{x}^*, R_{\mathsf{t}}^*) \leftarrow \mathcal{A}^{O_{\text{PC}}}(\mathsf{mpk})$

3 : **if** $\mathcal{ZK}.\mathsf{Vf}(\mathbf{R_L}, \mathsf{crs}, \pi^*, \mathsf{x}^*) == 1 \ \wedge \ q_D < q_S + q_R$: **return** 1

4 : **else return** 0

Figure 6.6: Security Games

Proof. For each payment request, the customer should transfer a tuple $\mathcal{T}[\pi, \mathsf{x}, R_{\mathsf{t}}, H(Tx)]$ where R_{t} is the auxiliary data at time slot t to convince the merchant and the group of statekeepers about the uniqueness of a collateral. Under the existence of a privacy-preserving blockchain Tx_{ID} does not reveal any information beyond the validity of the transaction and it protects the anonymity of the costumers. In this case, to prove that our construction preserves the anonymity of honest customers and provides unlinkability of payments we show that no PPT adversary, $\mathcal{A}$, by providing two pair of challenge secret keys/collateral keys $(\hat{\mathsf{sk}}_0^*, \hat{k}_0^*)$ and $(\hat{\mathsf{sk}}_1^*, \hat{k}_1^*)$, can distinguish between $(\pi_0, \mathsf{x}_0, R_{\mathsf{t},0})$ and $(\pi_1, \mathsf{x}_1, R_{\mathsf{t},1})$ as the output of the spending algorithm. This property is guaranteed because of the following main security properties for the given primitives: Zero-Knowledge property of the given NIZK proof system, computationally hiding property of the given commitment scheme, static-semantically secure property of the given randomness-reusable threshold encryption in bilinear groups and also the weak robustness of the given PRF.

Let the hybrid H^β be the case where the *Anonymity* experiment, $\text{EXP}^{\text{ANON}}_{\mathcal{A}}(\lambda, \beta)$ is run for $\beta = \{0,1\}$. In this case, we form a sequence of hybrids and show that each of the

successive hybrids are computationally indistinguishable from the preceding ones.

- **Hybrid H_1^{β}:** In this game, we assume the existence of an efficient simulator Sim and then modify the previous hybrid, H^{β}, by generating the challenge NIZK proof π_{β} via the simulation algorithm, $\pi'_{\beta} \leftarrow \mathcal{ZK}.\mathsf{Sim}(\mathsf{crs}, \hat{\mathsf{ts}}, \mathsf{x}_{\beta})$.

The Zero-Knowledge property of NIZK arguments provided in definition 22 guarantees that this experiment is indistinguishable from the one for H^{β} and we can write $H_1^{\beta} \approx_{\lambda} H^{\beta}$.

- **Hybrid H_{β}^2:** In this game, we modify H_1^{β} s.t. for generating the index *id* the challenger commits $\hat{\mathsf{sk}}_{1-\beta}^{*}$ instead of $\hat{\mathsf{sk}}_{\beta}^{*}$.

According to the hiding property of the given commitment scheme, this experiment is indistinguishable from H_{β}^1 and we can write, $H_2^{\beta} \approx_{\lambda} H_1^{\beta}$.

- **Hybrid H:** In this game, we modify H_{β}^2 by assuming the challenger runs the RRTE encryption algorithm under the message $m_{1-\beta}$ instead of m_{β}.

According to the Static Semantic Security property of the proposed randomness-reusable Threshold encryption, this experiment is indistinguishable from H_{β}^2. To be more concrete, $\mathcal{A}$ cannot distinguish between $\mathtt{Ct}_{\beta}$ and $\mathtt{Ct}_{1-\beta}$ as long as no twin ciphertext is generated even if the proofs are simulated. Thereby we have, $H_0 \approx_{\lambda} H_0^1 \approx_{\lambda} H_0^2 \approx_{\lambda} H \approx_{\lambda} H_1^1 \approx_{\lambda} H_1^2 \approx_{\lambda} H_1$.

To conclude this security property for the proposed construction, based on the weakly robust property of the given PRF, it is straightforward to demonstrate that the output of a PRF under two distinct keys is computationally indistinguishable and no PPT adversary can distinguish $R_{\mathsf{t},0}$ and $R_{\mathsf{t},1}$. □

□

Theorem 2. *The proposed generic construction in fig. 6.5 satisfies the payment certainty of payment guarantees as defined in definition 2.*

Proof. We prove this security property by contradiction and for the simplicity we avoid the hat notion for the secret parameters. Let there is a PPT adversary $\mathcal{A}$ that can break the payment certainty of the scheme and pass the verification phase without meeting at least of the following cases.

- **Case 1.** The adversary $\mathcal{A}$ can forge a valid payment guarantee, $\mathcal{T}$.
- **Case 2.** The adversary $\mathcal{A}$ can forge a valid aggregated signature σ_{R_t} s.t. $|\sigma_{R_t}| \geq (|\mathsf{StK}|/2)+1$.

By relying on the existence of a weakly-robust PRF, a *Knowledge Sound* NIZK argument, an existentially unforgeable TSPS construction we show that the success probablity of adversary in "**Case 1**" is negligible. Thus having played a sequence of indistinguishable games between $\mathcal{B}^{\text{PRF}}_{\text{WR}}(1^\lambda), \mathcal{B}^{\text{TSPS}}_{\text{EUF-CIMA}}(1^\lambda), \mathcal{B}^{\text{NIZK}}_{\text{KS}}(1^\lambda)$ and a PPT adversary $\mathcal{A}$, we gradually turn the payment certainty security game into the security features of the underlying primitives.

- **Game** G_0**:** In the first security game, let $\mathcal{A}$ forms a challenge transaction τ^* such that $\sum \mathcal{L}_c + \tau^* > \mathsf{col}_{\mathcal{A}}$ return a valid pair (π^*, x^*) with a non-negligible advantage ε. By contradiction, we assume $\mathcal{A}$ can win this game with a non-negligible advantage ε and we can write, $Adv^{\mathsf{PC}}_{\mathcal{A}}(\lambda) = \Pr[\mathcal{A} \text{ Wins } G_0] \geq \varepsilon$.
- **Game** G_1**:** In this game, we modify G_0 such that we assume the existence of an efficient extractor $\mathsf{Ext}(.)$. In this case, there exists an extractor that takes the extraction trapdoor $\hat{te}$ and the received challenge tuple (π^*, π^*_j, x^*) as inputs, and returns the corresponding witness $(w^*) \leftarrow \mathsf{Ext}(te, x^*, \pi^*)$ s.t. $w^* = \Big(\mathsf{cert}^*, \mu^*, \mathsf{sk}^*_c, r^*_t, (M^*_j, k^*)\Big)$. To be more precise, the extractor first extracts the indexed DH message $M^*_j := (id^*_j, M^*_{j1}, M^*_{j2})$, and then can extracts the secret PRF key k^* from the proof (π^*_j) as a proof to show the well-formedness of the index id^*_j. The indistinguishability of G_0 and G_1 can be proven via the *Knowledge Extraction* property of NIZK arguments, defined in definition 23. This property guarantees the existence of the defined extractor under non-falsifiable assumptions and we can write, $Adv^{\mathsf{PC}}_{\mathcal{A}}(\lambda) = \Pr[\mathcal{A} \text{ Wins } G_0] \approx \Pr[\mathcal{A} \text{ Wins } G_1]$ and this advantage consequently depends on two possible cases,

  $$\Pr[\mathcal{A} \text{ Wins } G_1] = \Pr[\mathcal{A} \text{ Wins } G_1 : (w^*, x^*) \in \mathbf{R_L}] + \Pr[\mathcal{A} \text{ Wins } G_1 : (w^*, x^*) \notin \mathbf{R_L}] \ .$$

  The probability of an adversary in the latter case can be bounded by the advantage a NIZK's knowledge soundness.

  $$Adv^{\mathsf{PC}}_{\mathcal{A}}(\lambda) \leq \Pr[\mathcal{A} \text{ Wins } G_1 : (w^*, x^*) \in \mathbf{R_L}] + Adv^{\text{NIZK}}_{\mathcal{B}_{\text{KS}}}(\lambda) \, .$$

  Under the assumption that the given NIZK is KS, the adversary $\mathcal{A}$ can win the game when the event of $(w^*, x^*) \in \mathbf{R_L}$ occurs.
- **Game** G_2**:** The challenger for the payment certainty security game can modify G_1 to an attacker against the weakly-robust PRF security game. The intended

key k^* is either a valid key $k^* \in \mathcal{K}$ s.t. it is not corrupted by the adversary, i.e. $k^* \notin \mathcal{CL}'$ or it is generated under a random key $k^* \notin \mathcal{K}$. The latter case will be bounded by the advantage of $\mathcal{B}^{\text{PRF}}_{\text{WR}}(1^\lambda)$ attacker, then we can write,

$$\Pr[\mathcal{A} \text{ Wins } G_2] = \Pr[\mathcal{A} \text{ Wins } G_2 : k^* \in \mathcal{K} \wedge k^* \notin \mathcal{CL}'] +$$

$$\Pr[\mathcal{A} \text{ Wins } G_2 : k^* \notin \mathcal{K}] \leq \Pr[\mathcal{A} \text{ Wins } G_2 : k^* \in \mathcal{K} \wedge k^* \notin \mathcal{CL}'] + Adv^{\text{PRF}}_{\mathcal{B}_{\text{WR}}}(\lambda) \ .$$

- **Game G_3:** This is the game G_2, except for a valid pair of witness and statement in $\mathbf{R_L}$ and a fresh and not queried PRF key, one can reduce it to a forgery attack for the underlying TSPS scheme. More specifically, if $k^* \in \mathcal{K}$ and not corrupted before then the challenger can generate its iDH message format M^*_j. Lets the set of authorities indices that are queried before to get a certificate by the adversary is denoted by $\mathcal{S}_{(\star,M^*_{j2})}$. If $|\mathcal{S}_{(\star,M^*_{j2})} \cup \mathcal{AU}'| < t$, $\mathcal{B}^{\text{TSPS}}_{\mathsf{EUF\text{-}CiMA}}(1^\lambda)$ returns the pair (M^*_j, cert^*) as a valid forgery for the defined threshold EUF-CiMA security game in Def. [17, 4.3]. Thus, we can write,

$$Adv^{\mathsf{PC}}_{\mathcal{A}}(\lambda) \leq Adv^{\text{NIZK}}_{\mathcal{B}_{\text{KS}}}(\lambda) + Adv^{\text{PRF}}_{\mathcal{B}_{\text{WR}}}(\lambda) + \Pr[\mathcal{A} \text{ Wins } G_3 : |\mathcal{S}_{(\star,M^*_{j2})} \cup \mathcal{AU}'| < t] +$$

$$\Pr[\mathcal{A} \text{ Wins } G_3 : |\mathcal{S}_{(\star,M^*_{j2})} \cup \mathcal{AU}'| \geq t] \leq Adv^{\text{NIZK}}_{\mathcal{B}_{\text{KS}}}(\lambda) +$$

$$Adv^{\text{PRF}}_{\mathcal{B}_{\text{WR}}}(\lambda) + Adv^{\text{TSPS}}_{\mathcal{B}_{\mathsf{EUF\text{-}CiMA}}}(\lambda) + \Pr[\mathcal{A} \text{ Wins } G_3 : |\mathcal{S}_{(\star,M^*_{j2})} \cup \mathcal{AU}'| \geq t] \ .$$

Since it is assumed that the adversary $\mathcal{A}$ should provide a fresh and not queried collateral, the probability of the event, "$\mathcal{A}$ Wins $G_3 \wedge |\mathcal{S}_{(\star,M^*_{j2})} \cup \mathcal{AU}'| \geq t$", is equal to zero. Then we can write,

$$Adv^{\mathsf{PC}}_{\mathcal{A}}(\lambda) \leq Adv^{\text{NIZK}}_{\mathcal{B}_{\text{KS}}}(\lambda) + Adv^{\text{PRF}}_{\mathcal{B}_{\text{WR}}}(\lambda) + Adv^{\text{TSPS}}_{\mathcal{B}_{\mathsf{EUF\text{-}CiMA}}}(\lambda) \ .$$

Similarly to demonstrate that the probability of **Case 2** is negligible we rely on the unforgeability of the given aggregatable digital signature. If the adversary $\mathcal{A}$ be able to forge a valid aggregated signature for a majority of the statekeepers then as it is assumed it only can corrupt at most $|\mathcal{M}'| < |\mathsf{StK}|/2$, then we can form an efficient algorithm $\mathcal{B}^{\text{DS}}_{\text{EUF-CMA}}(1^\lambda)$ to break the EUF-CMA property of the underlying DS scheme. Then we can write:

$$Adv^{\mathsf{PC}}_{\mathcal{A}}(\lambda) \leq Adv^{\text{NIZK}}_{\mathcal{B}_{\text{KS}}}(\lambda) + Adv^{\text{PRF}}_{\mathcal{B}_{\text{WR}}}(\lambda) + Adv^{\text{TSPS}}_{\mathcal{B}_{\mathsf{EUF\text{-}CiMA}}}(\lambda) + Adv^{\text{DS}}_{\mathcal{B}_{\text{EUF-CMA}}}(\lambda) \ .$$

Thus, as long as the underlying primitives are knowledge sound, weakly robust and existentially unforgeable then we can conclude the theorem. □ □

4 An Efficient Instantiation

In this section, we specify the concrete cryptographic primitives used to instantiate our construction. With the exception of RRTE, which is our novel construction, we refer formal definitions of primitives and their security properties to appendix A.1. We would like to stress the modularity of our construction. The below tools are used in a black box manner and can be replaced by superior tools that future research will inevitably develop.

Randomness-Reusable Threshold Encryption. (ℓ,t,k)-RRTE is a new observation on threshold encryption (TE) schemes (see Appendix A.1.5) and enables plaintext confidentiality as long as less than k number of ciphertexts with the same randomness is generated. Once a data owner issues at least k supplementary ciphertexts, it is publicly retrievable and everybody can blind out the encrypted data. We formulate this primitive for compatibility with the rest of our system, but it is worth noting that the underlying idea is similar to offline double spending detection used in e-cash schemes.

Definition 3 (Randomness-Reusable Threshold Encryption)**.** *For a given public parameters* pp *and security parameter* λ*, a* (ℓ,t,k)*-RRTE,* Ψ_{RRTE}*, over the message space* $\mathcal{M}$ *and ciphertext space* $\mathcal{C}$ *consists of three main PPT algorithms defined as follows:*

- $(\mathsf{pk},\mathsf{pk}) \leftarrow \mathcal{RRTE}.\mathsf{KGen}(\mathsf{pp},\ell,t,k)$*: Key generation is a probabilistic and distributed algorithm that takes* pp *along with three integers* $\ell,t,k \in \mathsf{poly}(\lambda)$ *as inputs. It then returns a vector of public key* pk *of size* ℓ *and a general public key* pk *as outputs.*
- $(\mathtt{Ct}_j, v) \leftarrow \mathcal{RRTE}.\mathsf{Enc}(\mathsf{pp},\mathsf{pk},m,\mathsf{pk}_j)$*: The encryption algorithm as a probabilistic algorithm takes* pp*, global public key* pk*, a message* $m \in \mathcal{M}$ *along with a public key* pk_j *as inputs. It returns ciphertext* $\mathtt{Ct}_j \in \mathcal{C}$ *associated with the recipient* $j \in \mathcal{R}$ *and an auxiliary value* v *as outputs.*
- $(\perp, m) \leftarrow \mathcal{RRTE}.\mathsf{Dec}(\mathsf{pp},\{\mathtt{Ct}_j\}_{j\in\mathcal{K}},v)$*: The decryption algorithm takes* pp*, a set of ciphertexts* $\{\mathtt{Ct}_j\}_{j\in\mathcal{K}}$ *along with an auxiliary value* v *as inputs. If* $|\mathcal{K}| \geq k$*, it returns* $m \in \mathcal{M}$*, else it responds by* $\perp$*.*

Note that in Appendix A.1.5, we elaborate more on the security requirements and then we propose an efficient construction based on threshold ElGamal encryptions.

Pseudo-Random Function. We utilise a weakly-robust PRF proposed by Dodis and Yampolskiy [19] in order to make customers' collaterals reusable. This PRF enables us to define a time of payment, i.e. x in $\mathsf{PRF}_k(x) = g^{1/(k+x)}$ function and prove the validity of operations, efficiently. This ensures that a customer always has to input the time of payment, which is then verified by a receiving merchant. By utilizing this property and its combination with RRTE, as discussed in section 1, we can block a customer from reusing the same collateral for a pre-defined time period.

Digital Signature Schemes. We require two types of signatures, one for the authorities and another for the statekeepers. These signatures need non-overlapping properties which we detail below.

- Threshold Structure-Preserving Signatures [17]. There are two reasons to use the TSPS scheme proposed by Crites et al. Firstly, like any other digital signature, it provides authentication, such that no entity except the qualified authorities can issue collateral proofs. Secondly, due to its threshold nature, TSPS enables our construction to rely on an honest majority (authorities) instead of a central trusted party.
- BLS Signatures [9]. BLS Signatures are efficiently aggregatable, and thus they are useful in our setting. A statekeeper must validate payment requests from various merchants, and they do so using BLS signatures. For a victim merchant to redeem user collateral from the smart contract, they may first aggregate the signatures allowing for a shorter interaction.

Commitment Scheme. In our construction, we use the Pedersen commitment scheme [39] due to the following reasons; firstly, the TSPS construction is defined over the indexed DH message spaces (Definition 5) and each secret PRF key needs to get an index. Hence, these commitments are used to the secret scalar PRF keys as an index. Secondly, the hiding property of such commitments masks the secret PRF keys used in our construction. In addition, the binding property of Pedersen commitments ensures the unforgeability of these secret PRF keys. Finally, Pedersen commitments are compatible with discrete logarithm-based proofs like original Sigma protocols and enables customers to efficiently prove knowledge of these committed values.

NIZK Proofs. To instantiate the described NP-relations in section 3.1, we utilize three main proof systems: Sigma protocols [41], range-proofs [11] and GS proof systems [26] (see Appendix A.1.7). Sigma protocols are an efficient choice as the main proof system in our implementation; we use the Fiat-Shamir heuristic [24] to make then non-interactive. Range-proofs enable us to prove that a hidden value lies in a range interval. GS proof systems are useful as they are secure in the standard model and support a straight-line extraction of the witnesses. Additionally, the instantiation

of these proofs does not require any trusted setup and can be batched: this enables an efficient verification [30].

5 Performance Analysis

In this section, we demonstrate the performance of our system. Based on the application, the costs incurred in each phase are divided into two parts, termed "offline phase" and "online phase". The former includes the parameter generation, key generation and registration functions. The latter is solely responsible for spending and verification and is the main focus of this evaluation.

Our experimental setup is similar to the one in Snappy. Namely, we distribute various parties in different regions around the world and measure the end to end latency of transactions.[2]

Specifically, our implementation uses the Charm-Crypto framework [1], a Python library for Pairing-based Cryptography and obtained the benchmarks on four AWS EC2 instances. The scenario we consider is similar to the one in Snappy. Namely, merchants, customers and statekeepers are globally distributed in four different locations and we create 1 000 tps in order to measure the average time it takes for one transaction to complete. Since transactions are distributed to many merchants and the merchants run independently, it is possible to create an equivalent scenario and only consider the work needed for one merchant. Consider the workload from the perspective of a single statekeeper: its workload depends on transactions that are passing through all other merchants. To accurately estimate the workload of a single statekeeper, we injecting artificial verification requests to it. Our scenario is summarized in fig. 6.7.

All our EC2 instances had the same computational configuration, i.e., an Ubuntu Server 20.04 LTS (HVM) with an Intel (R) Xeon(R) CPU @ 2.50 GHz and 16 GB of memory. We apply the Barreto-Naehrig (BN254) curve (also known as type F groups), $y^2 = x^3 + b$ with embedding curve degree 12 [6]. In this pairing group, the base field order is 256 bits.

Latency. As illustrated in fig. 6.8, the latency for each transaction grows linearly with the number of statekeepers verifying this transaction (depicted with the orange line). During our evaluations, we noticed that the time required for a customer to generate and send a payment guarantee (depicted with the red line) is mostly constant, i.e., $\sim$ 240 ms. However, in the case of 40 statekeepers, our construction allows a customer and merchant to successfully transact within $\sim$ 550 milliseconds (ms). In contrast, in the case of 200 statekeepers, a transaction takes $\sim$ 1.3 seconds (s). Hence, the time required to guarantee a payment to merchants in our construction is bounded

[2]The open-source implementation can be found in this repository.

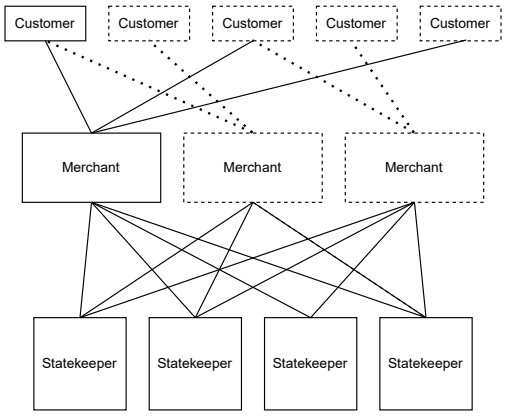


Figure 6.7: Experimental setup. Dotted lines and shapes represent parties and workload that do not affect the leftmost customer or the leftmost Merchant. Our experiment only considers the solid lines and shapes. From the perspective of the leftmost client and the leftmost merchant, this is equivalent to running the full system. While statekeepers are the same as merchants, we make these two sets distinct for clarity

by the set of statekeepers. A direct comparison with Snappy is not possible for two reasons; firstly, the evaluation of Snappy only considers the time taken for payment approval, i.e., it does not consider the time spent on customer-merchant interaction. Secondly, their simulation code is not freely available. A standalone analysis shows that our construction provides privacy against statekeepers without impacting the latency of payments. Greater number of statekeepers provides more robustness and better protection against double-spends, but requires stronger liveness assumptions on top of increasing the latency. We find 120 statekeepers to be an optimal trade-off of these factors, ultimately leading to latency lower than 1 sec.

Smart Contract Cost. The transition between states happens depending on the function calls on the smart contract. For simplicity, we describe here only the functionality of the smart contract focusing on one customer and multiple merchants. We refer to our smart contract as $Auth_{SC}$, an entity is referred to as e_x where $x = c$ or m for customer or merchant respectively. The underlying ledger is referred to as L_{SC} and an entity's account on that ledger is referred to as Acc_L^x where $x = c$ or m respectively. The private ledger of the merchants is referred to as $Bull_m$, since it behaves like a bulletin board. $Auth_{SC}$ has seven states as follows:

init: $Auth_{SC}$ is deployed. e_c can now deposit funds (col_c). If so, then change state to *ready*. Else do not change state.

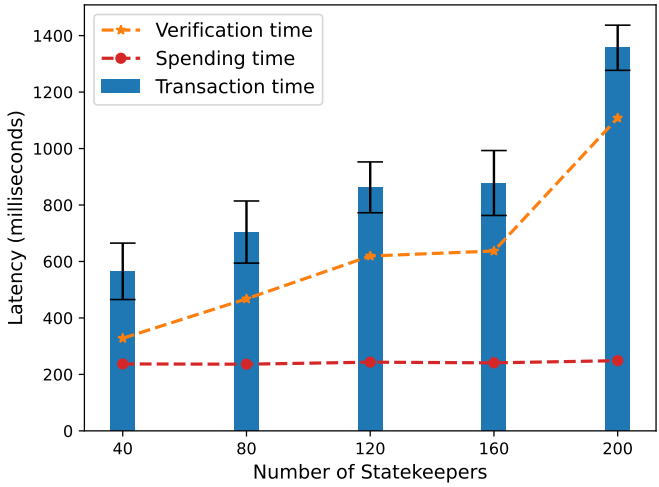


Figure 6.8: Latency comparison. Total transaction time for 1000 tx per second vs the number of statekeepers

ready: e_c successfully registers by depositing col_c in $Auth_{SC}$. If col_c is available in $Auth_{SC}$, change state to *pay*. Else do not change state.

pay: If e_c has made payment (pay_{m_i}), change state to $reclaim_m$. Else do not change state.

$reclaim_m$: Check $Bull_m$ for double-spends from e_c. If double-spend present, use secret to reclaim pay_{m_i} and change state to *withdraw*. If no double-spend found until actual payment received, change state to $reclaim_c$.

$reclaim_c$: If 1 day has passed since pay_{m_i}, reclaim pay_{m_i} and add it to col_c. Then, change state to *withdraw*. Else do not change state.

withdraw: If e_c wants to exit the system and the state is *withdraw* or *ready*, send money from $Auth_{SC}$ to Acc_L^c and change state to *exit*. If e_w wants to exit the system and the state is *withdraw*, send money from $Auth_{SC}$ to Acc_L^w and change state to *exit*. Else do not change state.

exit: Remove e_x from $Auth_{SC}$ and change state to *init*.

Table 6.1 lists the gas fees of executing various functions of our system. In the first column, we mention the amount of base gas fee, and then we express it as a proportion

of minimum gas fee of a transaction. The minimum gas fee is set to 21 000 GWei, which is also the amount of gas that standard on-chain payments require. Note that the table reflects the fees users can expect to pay, although the actual amount also depends on the so-called priority fee which depends on the current traffic in the Ethereum transaction market. More information about the costs incurred due to our SC is given in section 6.

Table 6.1: Costs of transactions on our smart contract deployed on Ethereum, and the cost as a proportion of a standard transaction $\Delta = 21\,000$.

Function	Customer registration	Pay	Merchant registration	Reclaim	Withdraw Balance
Gas	107 400	44 055	54 317	34 972	22 352
$\times\Delta$	5.1	2.09	2.59	1.65	1.06

6 Discussion

Collateral reusability. The capital demands of existing layer-2 proposals have been frequently acknowledged [40, 37, 18, 36, 28, 29, 33]. Roughly speaking, the risk the system is willing to take on behalf of a user is proportional to the amount of money the user commits as collateral. This limitation is difficult to overcome in a trustless setting, and our work can be seen as partial progress in this direction.

Payment channel networks have heavy collateral requirements in order to maintain trustlessness. A payment of, say, 10 dollars across 3 channels requires committed 10 dollar's worth of collateral on each of the three channels. These funds are locked up for a fixed amount of time and cannot be used for other payments. In early works, the funds were locked up for a time proportional to the length of the path used, but recent advances have reduced this to an amount depending only the security parameter and block generation time [2]. Despite this recent innovation, we find the collateral requirements prohibitive for an efficient payment scheme.

Payment channels, if used in the same direction, also suffer from collateral depletion, rendering further payments in that direction impossible. This problem of channel depletion has somewhat been addressed with re-balancing techniques [4]. However, rebalancing is generally not possible for a user that only employs their channels for payments and not for getting paid. Hence, rebalancing is incompatible with retail payments which are generally unilateral i.e., from the set of consumers to the set of merchants.

In our system, payment guarantees assure a payee (merchant) that they will receive a payment once the transaction on the underlying blockchain has reached its eventual finality, even if the payer (customer) is malicious. The collateral provided by a payer is *reusable*: this reusability comes from the abstraction of payment guarantees, where the customer makes a payment on the underlying blockchain and assures the payee about this payment by using irrefutable evidence that this payment will be successful. Unlike existing systems like Snappy, where reusability of collaterals comes at the cost of privacy, we utilize RRTE to guarantee privacy for all honest customers. As long as customers don't attempt to double spend, that is, simultaneously issue multiple payment guarantees with the same collateral, their identities are kept secret.

Trust Assumptions. There is a general trade-off between the efficiency and privacy of a financial system and the level of trust assumed between participants. For instance, a trusted central entity can efficiently set up a digital currency system, as evidenced by Chaumian e-cash. Measures of transaction latency and throughput thrive at the high cost of trust in the central authority. On the other end, decentralized blockchains achieve functional but slow financial systems without requiring trust in any single party.

In payment channel networks, a similar trade-off has been observed by Avarikioti et al. [3], who suggest that PCNs are more stable and efficient when centralized structures are present. In an empirical survey, Zabka et al. [42] observe the rising centrality in the Lightning Network as the capacity and capabilities of Lightning grew over time.

To the best of our knowledge, we are the first ones to utilize these building blocks and design an instant finality layer-2 scheme that utilizes *reusable* collaterals, while offering several privacy-preserving properties. In order to offer these properties, we need to rely on a set of trust assumptions. Our architecture is semi-decentralized in the sense that we rely on an honest majority of authorities to initialize our construction. This is similar to the approach of LDSP [38]. However, unlike LDSP, our authorities do not play any role in our payment protocol. The merchants and statekeepers have greater power to punish dishonest customers by confiscating their collateral. Yet, we allow an honest majority of merchants to do so *only* against customers who attempt to double-spend, not honest customers. Moreover, the design is permissionless in that cryptocurrency holders can freely participate as customers. Considering the general trade-off between centrality, trust, performance and efficiency, we consider our setup to lie in a "sweet spot." In our case, this balance was achieved through advances in cryptography, namely RRTE. We call for further cryptographic innovations and welcome research into even more trustless, robust and secure systems.

Privacy. The main idea underlying our private double-spending protection, goes all the way back to the e-cash schemes introduced by Chaum [15]. "On-line" e-cash tackled the problem of double spending by having the issuing bank verify each transaction individually before it was marked successful. Chaum, Fiat and Naor later extended this idea to support "off-line" payments, i.e, a customer could make untraceable payments

to the merchant without involving a bank for every transaction [16]. Due to this "off-line" nature of payments, the solution for double-spending combined prevention with tamper-resistant hardware along with detection through successful tracking, i.e., the issuing bank would check the list of all coins spent, and once double-spending was spotted the identity of the perpetrator would be revealed. This approach of realizing offline payments while detecting double-spending, known as the Chaum-Fiat-Naor (CFN) approach was adopted and improved by several following e-cash systems [10, 25, 13, 14, 5]. The existing plethora of literature has made several improvements to Chaum's e-cash, however, all work with centrally issued currency and mostly rely on a custodian bank to catch double-spending. Our work is also an application of the CFN approach, with the major difference of building upon decentralized cryptocurrencies for safely reducing latency, instead of building an entire e-cash scheme from the ground up. When compared to double-spend detection techniques in e-cash (for instance the one recently used in [7]), our novel RRTE is more efficient in terms of communication rounds, allows the deposited collaterals to be reused and by default enables anyone to track double-spends.

However, *on-chain privacy* is derived from the underlying blockchain, and is the highest level of privacy that one can hope to achieve at the protocol level. In other words, implementing our overlay on a completely de-anonymized and public blockchain cannot make the payments private, since the underlying blockchain will reveal private data no matter how secure the protocol. Similarly, developing on top of private blockchains such as Monero doesn't directly solve the privacy issues of earlier works that allowed transactions of the same user to be linked.

In this way, the question of blockchain-level privacy is relevant yet orthogonal to our work. While Snappy claims that future improvements such as deployment on privacy-preserving blockchains that support privacy-preserving SC will enable their construction to provide on-chain privacy, that is not true. As explained in Section 2.1, an inherent design flaw in their proactive double-spending detection is overlooked.

Our construction provides this proactive double-spending detection while guaranteeing protocol-level privacy. Hence, deployment on a privacy-preserving blockchain makes our construction fully private, even on-chain. There are possible workarounds such as change addresses and mixing services to provide increased on-chain pseudonymity in case of public blockchains. Achieving a greater synergy between these two levels of a privacy seems to be a promising avenue for future work.

On-chain Transaction Fees. The transaction fees in our system differ from conventional fees since the customer pays the merchant indirectly through a smart contract (SC). This is necessary to prevent an on-chain double-spend by a malicious customer. By on-chain double-spend, we mean to distinguish between a double-spend attempt of customer collateral, and a double-spend on the underlying blockchain itself. Even if a malicious customer can influence miners and induce a blockchain

double-spend, the SC-based transaction is able to remunerate the affected merchant. Simply put, the SC can escrow the funds until sufficient confirmations of on-chain payment have been found. We stick to the convention of 6 succeeding blocks after said transaction.

As discussed in section 5, executing payment through our SC incurs twice the on-chain transaction fees of a standard on-chain Ethereum payment. There is an additional fee incurred by the merchant during withdrawal of payments, but this is far less frequent than the former. Nevertheless, it is desirable to construct a more cost efficient yet secure system for direct customer to merchant payments.

A potential fix could be to encode specific spend conditions for user collaterals. To be precise, any merchant can move the collateral by providing evidence of a conflicting transaction on the blockchain. This could be implemented via a payment guarantee to said merchant, along with on-chain evidence of a conflicting payment. We leave this implementation, along with other possible optimizations of fees, for future work.

Incentives of Involved Parties. This work deals with the cryptographic challenges of achieving privacy while reducing latency of cryptocurrency payments. Our focus is admittedly myopic, as we overlook practical aspects of incentives. For instance, we refer to authorities that register merchants and customers, but these authorities lack a concrete incentive to fulfil this role honestly. As an initial and arbitrary choice, we selected a subset of involved merchants to play this role of authorities, while requiring an honest majority of authorities.

It is unclear who should be playing this role, and what their incentives should be. Could we perhaps allocate a small fee per merchant to authority? Or automatically grant a fraction of collaterals confiscated from dishonest users? Or even eliminate this issue entirely by building more advanced cryptography so that our overlay can be set up even without their existence?

Similarly, we lack a clear explanation of incentives for statekeepers. A basic solution would be to allocate a certain fraction of each transaction value to the statekeepers; however, this still needs to be properly analyzed in order to confirm if such an incentive is sufficient.

7 Conclusion

In this paper, we present a new overlay network for instant confirmation of cryptocurrency transactions, that also maintains anonymity of users and unlinkability of their transactions. On the one hand, it allows merchants in a retail system to safely accept fast payments without risk of double-spending. On the other, dishonest customers who attempt to double-spend get their identities exposed and their collateral

confiscated to reimburse the merchants. Honest customers, however, are able to *reuse* their collaterals.

To this end, we designed a novel randomness-reusable threshold scheme, that enables participants to audit the payments in the network and reveal the identity of malicious customer who perform double-spending. This threshold encryption scheme maintains the privacy of honest customers who do not attempt to double-spend. We provide a formal proof of security with respect to three main features namely *customers' anonymity, unlinkability of transactions* and *payment certainty for merchants*. We motivate our choice of cryptographic primitives and efficiently implement them. Our evaluation shows that our construction allows for fast global payments with a delay of less than 1 seconds.

References

[1] Joseph A. Akinyele, Christina Garman, Ian Miers, Matthew W. Pagano, Michael Rushanan, Matthew Green, and Aviel D. Rubin. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3(2):111–128, June 2013.

[2] Lukas Aumayr, Kasra Abbaszadeh, and Matteo Maffei. Thora: Atomic and privacy-preserving multi-channel updates. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 165–178, New York, NY, USA, 2022. Association for Computing Machinery.

[3] Zeta Avarikioti, Lioba Heimbach, Yuyi Wang, and Roger Wattenhofer. Ride the lightning: The game theory of payment channels. In *FC 2020*, LNCS, pages 264–283. Springer, Heidelberg, 2020.

[4] Zeta Avarikioti, Krzysztof Pietrzak, Iosif Salem, Stefan Schmid, Samarth Tiwari, and Michelle Yeo. HIDE & SEEK: Privacy-preserving rebalancing on payment channel networks. Cryptology ePrint Archive, Report 2021/1401, 2021. https://eprint.iacr.org/2021/1401.

[5] Foteini Baldimtsi, Melissa Chase, Georg Fuchsbauer, and Markulf Kohlweiss. Anonymous transferable E-cash. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 101–124. Springer, Heidelberg, March / April 2015.

[6] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford Tavares, editors, *SAC 2005*, volume 3897 of *LNCS*, pages 319–331. Springer, Heidelberg, August 2006.

[7] Balthazar Bauer, Georg Fuchsbauer, and Chen Qian. Transferable E-cash: A cleaner model and the first practical instantiation. LNCS, pages 559–590. Springer, Heidelberg, 2021.

[8] Yaacov Benmeleh. Blockchain firm starkware valued at $2 billion in funding round. https://www.bloomberg.com, 2021.

[9] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004.

[10] Stefan Brands. Untraceable off-line cash in wallets with observers (extended abstract). In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 302–318. Springer, Heidelberg, August 1994.

[11] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.

[12] Vitalik Buterin. An incomplete guide to rollups. https://vitalik.ca/general/2021/01/05/rollup.html, 2021.

[13] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 302–321. Springer, Heidelberg, May 2005.

[14] Sébastien Canard and Aline Gouget. Multiple denominations in e-cash with compact transaction data. In Radu Sion, editor, *FC 2010*, volume 6052 of *LNCS*, pages 82–97. Springer, Heidelberg, January 2010.

[15] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982.

[16] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 319–327. Springer, Heidelberg, August 1990.

[17] Elizabeth Crites, Markulf Kohlweiss, Bart Preneel, Mahdi Sedaghat, and Daniel Slamanig. Threshold structure-preserving signatures. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023*, pages 348–382, Singapore, 2023. Springer Nature Singapore.

[18] Christian Decker and Roger Wattenhofer. A fast and scalable payment network with Bitcoin duplex micropayment channels. In *Symposium on Self-Stabilizing Systems*, pages 3–18. Springer, 2015.

[19] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In Serge Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 416–431. Springer, Heidelberg, January 2005.

[20] Stefan Dziembowski, Lisa Eckey, Sebastian Faust, and Daniel Malinowski. Perun: Virtual payment hubs over cryptocurrencies. In *2019 IEEE Symposium on Security and Privacy*, pages 106–123. IEEE Computer Society Press, May 2019.

[21] Electric Coin Company. Explaining viewing keys. https://electriccoin.co/blog/explaining-viewing-keys/. Accessed: 2023-02-13.

[22] Ethereum.org. Layer 2 rollups. Available in this Link, 2021.

[23] ethos.dev. The beacon chain Ethereum 2.0 explainer you need to read first. https://ethos.dev/beacon-chain. Accessed: 2023-05-10.

[24] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

[25] Yair Frankel, Yiannis Tsiounis, and Moti Yung. "indirect discourse proof": Achieving efficient fair off-line E-cash. In Kwangjo Kim and Tsutomu Matsumoto, editors, *ASIACRYPT'96*, volume 1163 of *LNCS*, pages 286–300. Springer, Heidelberg, November 1996.

[26] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, 2008.

[27] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. SoK: Layer-two blockchain protocols. In *FC 2020*, LNCS, pages 201–226. Springer, Heidelberg, 2020.

[28] Mike Hearn. Micro-payment channels implementation now in bitcoinj. Bitcointalk.org, 2013.

[29] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. Tumblebit: An untrusted Bitcoin-compatible anonymous payment hub. In *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017*. The Internet Society, 2017.

[30] Gottfried Herold, Max Hoffmann, Michael Klooß, Carla Ràfols, and Andy Rupp. New techniques for structural batch verification in bilinear groups with applications to groth-sahai proofs. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1547–1564. ACM Press, October / November 2017.

[31] Kashmir Hill. How Target figured out a teen girl was pregnant before her father did. https://www.forbes.com/sites/kashmirhill/2012/02/16/how-target-figured-out-a-teen-girl-was-pregnant-before-her-father-did/?sh=53d927356668. Accessed: 2022-08-30.

[32] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *Int. J. Inf. Secur.*, 1(1):36–63, aug 2001.

[33] Rami Khalil, A Zamyatin, G Felley, P Moreno-Sanchez, and A Gervais. Commit-chains: Secure, scalable off-chain payments. Technical report, Cryptology ePrint Archive, Report 2018/642, 2018.

[34] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019.

[35] Vasilios Mavroudis, Karl Wüst, Aritra Dhar, Kari Kostiainen, and Srdjan Capkun. Snappy: Fast on-chain payments with practical collaterals. In *NDSS 2020*. The Internet Society, 2020.

[36] Andrew Miller, Iddo Bentov, Surya Bakshi, Ranjit Kumaresan, and Patrick McCorry. Sprites and state channels: Payment networks that go faster than lightning. In Ian Goldberg and Tyler Moore, editors, *FC 2019*, volume 11598 of *LNCS*, pages 508–526. Springer, Heidelberg, February 2019.

[37] Network, Raiden. What is the raiden network. https://raiden.network/101.html, 2019.

[38] Lucien K. L. Ng, Sherman S. M. Chow, Donald P. H. Wong, and Anna P. Y. Woo. Ldsp: Shopping with cryptocurrency privately and quickly under leadership. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 261–271, 2021.

[39] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.

[40] Joseph Poon and Thaddeus Dryja. The Bitcoin lightning network: Scalable off-chain instant payments. https://lightning.network/lightning-network-paper.pdf/, January 2016.

[41] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, August 1990.

[42] Philipp Zabka, Klaus-T. Foerster, Christian Decker, and Stefan Schmid. Short paper: A centrality analysis of the lightning network. In *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers*, page 374–385, Berlin, Heidelberg, 2022. Springer-Verlag.

Appendix A

Appendix

A.1 Cryptographic Primitives

In this section, we recall the needed preliminaries and definitions and then review the required building blocks, which include pseudo-random functions, Shamir secret sharing, aggregatable digital signature, Threshold Structure-Preserving Signatures, commitments and non-interactive zero-knowledge proofs. Finally, we also recall the definition of threshold encryptions as a starting point for the defined RRTE in section 4.

Definition 4 (Bilinear Groups [5])**.** *A* Type-III bilinear group generator[1] $\mathcal{BG}(1^\lambda)$ *returns a tuple* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{p}, e, g, h)$*, consisting of cyclic Abelian groups* $\mathbb{G}_1$*,* $\mathbb{G}_2$*, and* $\mathbb{G}_T$ *with the same prime order* p*. For given generators of groups* $\mathbb{G}_1$ *and* $\mathbb{G}_2$ *namely g and h, an efficient non-degenerate Type-III bilinear pairing is denoted by* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$*, such that,* $\forall\, a, b \in \mathbb{Z}_\mathsf{p} :\ e(g^a, h^b) = e(g^b, h^a) = e(g,h)^{ab}$ *and* $e(g,h) \neq 1_{\mathbb{G}_T}$.

Definition 5 (Indexed Diffie-Hellman Message Space [10])**.** *For a given bilinear group,* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{p}, e, g, h) \leftarrow \mathcal{BG}(1^\lambda)$*, and hash function* H *that is modeled in the random oracle,* $\mathcal{M}^{\mathsf{H}}_{\mathsf{iDH}}$ *is called the indexed DH message space, if we have:*

1. $\forall\, (id, M_1, M_2) \in \mathcal{M}^{\mathsf{H}}_{\mathsf{iDH}}\ \exists\, m \in \mathbb{Z}_\mathsf{p} : \boldsymbol{g} = \mathsf{H}(id), M_1 = \boldsymbol{g}^m, M_2 = h^m$.
2. *No two messages use the same index, i.e.,* $\forall\, (id, M_1, M_2) \in \mathcal{M}^{\mathsf{H}}_{\mathsf{iDH}}, (id', M_1', M_2') \in \mathcal{M}^{\mathsf{H}}_{\mathsf{iDH}}$**, if** $id = id' \Rightarrow (M_1, M_2) = (M_1', M_2') \in \mathcal{M}_{\mathsf{iDH}}$.

[1] No nontrivial homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ exists [16].

$\mathsf{iDH}^{\mathsf{H}}(id,m)$	$\mathsf{H}(id)$
1 : $\boldsymbol{g} \leftarrow \mathsf{H}(id)$	1 : **If** $Q_{\mathsf{H}}[id] = \perp$:
2 : $M_1 = \boldsymbol{g}^m$	2 : $r \leftarrow\$ \mathbb{Z}_{\mathsf{p}}$
3 : $M_2 = h^m$	3 : $Q_{\mathsf{H}}[id] \leftarrow g^r := \boldsymbol{g}$
4 : **return** $(id, M_1, M_2) \in \mathcal{M}^{\mathsf{H}}_{\mathsf{iDH}}$	4 : **return** $Q_{\mathsf{H}}[id]$

Figure A.1: Indexed Diffie-Hellman Message Space in the ROM

More formally, we recall the indexed DH message space generator in fig. A.1, such that $\mathsf{iDH}^{\mathsf{H}} : \mathbb{I} \times \mathbb{Z}_{\mathsf{p}} \rightarrow \mathbb{I} \times \mathbb{G}_1 \times \mathbb{G}_2$, where $\mathbb{I}$ is the index space.

A.1.1 Pseudo-Random Function (PRF)

Definition 6 (Pseudo-Random Functions [17])**.** *Let $\mathcal{K}$ and I be the key and input spaces, respectively. We say a family of functions like $f : \mathcal{K} \times I \rightarrow \mathcal{F}$ is a pseudo-random function (PRF) family if it is efficiently computable and for all PPT distinguishers $\mathcal{D}$ we have,*

$$\left| \Pr_{k \leftarrow\$ \mathcal{K}}[\mathcal{D}^{\mathsf{PRF}_k} \textit{ accepts}] - \Pr_{g \leftarrow\$ \mathcal{G}}[\mathcal{D}^{g} \textit{ accepts}] \right| \leq \mathsf{negl}(\lambda) \ ,$$

where $\mathcal{D}^O$ denotes the output of distinguisher $\mathcal{D}$ when given access to oracle O. It is assumed that the distinguisher can adaptively choose the inputs and $\mathcal{G} : I \rightarrow \mathcal{F}$ is a set of uniformly random functions.

We recall the weak notion of robustness for a PRF function, defined by Damgård et al. [11], such that no PPT adversary can find a key that produces collisions with a PRF generated by an honest key.

Definition 7 (Weakly-Robust PRF [11])**.** *A PRF scheme, Ψ_{PRF}, under query set $Q = (x,y) \in I \times \mathcal{F}$ is weakly-robust if for all PPT adversaries $\mathcal{A}$ we have:*

$$\Pr \begin{bmatrix} k \leftarrow\$ \mathsf{KGen}(\lambda), (x^*, k^*) \leftarrow\$ \mathcal{A}^{\mathsf{PRF}_k(.)}(1^\lambda) : \exists\ (x,y) \in Q, \\ \mathsf{PRF}_{k^*}(x^*) = y = \mathsf{PRF}_k(x) \end{bmatrix} \leq \mathsf{negl}(\lambda) \ .$$

In order to make customers' collaterals reusable, we utilise a weakly-robust PRF proposed by Dodis and Yampolskiy [12]. Let a cyclic group $\mathbb{G}$ of prime order p

with a generator g. The PRF on input $x \in \mathbb{Z}_\mathsf{p}$ under key space $\mathbb{Z}_\mathsf{p}$ is defined by $\mathsf{PRF}_k(x) = g^{1/(k+x)}$, where $k \leftarrow\$ \mathbb{Z}_\mathsf{p}$ is a secret key.

A.1.2 Shamir Secret Sharing

A (n,t)-Shamir Secret Sharing (SSS) [27] divides a secret s among n shareholders such that each subset of t shareholders can reconstruct secret s and any smaller subset of them learn nothing about the secret. For this purpose, the dealer who knows the secret s forms a polynomial $f(x)$ of degree $t+1$ with a randomly chosen coefficients such that $f(0) = s$. Then the dealer securely provides each shareholder with $s_i = f(i), i \in \{1,\ldots,n\}$. Particularly, each subset of $\mathcal{T} \subset \{1,\ldots,n\}$ with size at least t by pooling their shares can reconstruct the secret s using the Lagrange polynomial interpolation as $s = f(0) = \sum_{i\in\mathcal{T}} s_i L_i^{\mathcal{T}}(0)$, where $L_i^{\mathcal{T}}(x) = \prod_{j\in\mathcal{T}, j\neq i} \frac{x-j}{i-j}$.

A.1.3 Digital Signatures and TSPS

Digital signatures are an electronic analogue of written signatures that ensure data authentication, and the non-repudiation of the sender. Informally, an aggregatable signature over message space $\mathcal{M}$ consists of four main PPT algorithms: setup, key generation, signing and verification. On input of the security parameter, the setup algorithm generates the public parameters pp, the randomized key generation algorithm generates a pair of secret signing/verification key $(\hat{\mathsf{sk}}, \mathsf{vk})$. The signing algorithm generates a signature σ by taking a message $m \in \mathcal{M}$ along with pp and the secret signing key $\hat{\mathsf{sk}}$. The deterministic verification algorithm takes the verification key vk, a message m, and a signature σ as inputs and outputs 1 (accept) if the signature is valid and 0 (reject), otherwise.

Definition 8 (Digital Signature)**.** *For a given security parameter λ, a digital signature consists of the following PPT algorithms defined as follows:*

- $\mathsf{pp} \leftarrow \mathcal{DS}.\mathsf{Setup}(1^\lambda)$*: It takes the security parameter λ in its unary representation as input and outputs the set of public parameters* pp*.*
- $(\hat{\mathsf{sgk}}_i, \mathsf{vk}_i) \leftarrow \mathcal{DS}.\mathsf{KGen}(\mathsf{pp}, i)$*: This algorithm takes the global public parameters* pp *and an identification value i as inputs. It then returns the pair of signing/verification keys $(\hat{\mathsf{sgk}}_i, \mathsf{vk}_i)$ associated with a message space $\mathcal{M}$ and party i.*
- $\sigma_i \leftarrow \mathcal{DS}.\mathsf{Sign}(\mathsf{pp}, \hat{\mathsf{sgk}}_i, m)$*: On input of the signing key $\hat{\mathsf{sgk}}_i$ and a message $m \in \mathcal{M}$, this probabilistic algorithm outputs a signature σ_i.*

- $0/1 \leftarrow \mathcal{DS}.\mathsf{Vf}(\mathsf{pp}, \mathsf{vk}, \sigma, m)$*: This deterministic algorithm takes as inputs a set of verification keys* $\mathsf{vk} := \{\mathsf{vk}_i\}$ *for any* $i \in [w]$*, a signature* σ *and m and outputs either* 1 *(accept) or* 0 *(reject).*

Next we recall the primary security requirements for a digital signature: including *correctness* and *unforgeability against chosen message attack* and extendable to an aggregatable signature.

Definition 9 (Correctness)**.** *A digital signature scheme,* $\Psi_{\mathcal{DS}}$*, is called correct, if we have,*

$$\Pr\left[\forall\ (\hat{\mathsf{sgk}}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\mathsf{pp}), m \in \mathcal{M} : \mathsf{Vf}\left(\mathsf{pp}, \mathsf{vk}, m, \mathsf{Sign}(\mathsf{pp}, \hat{\mathsf{sgk}}, m)\right) = 1 \quad \right] \geq 1 - \mathsf{negl}(\lambda) \ .$$

Definition 10 (EUF-CMA)**.** *A digital signature,* $\Psi_{\mathcal{DS}}$*, is called* EUF-CMA*-secure if for all PPT adversaries* $\mathcal{A}$ *with an access to the signing oracle* $O_{\mathsf{Sign}}()$ *and the following winning advantage,* $Adv^{\text{EUF-CMA}}_{DS,\mathcal{A}}(\lambda)$*:*

$$\Pr\left[\begin{array}{l} \forall\ (\mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda), (\hat{\mathsf{sgk}}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\mathsf{pp}), \\ \\ (\sigma^*, m^*) \leftarrow\!\!\$\ \mathcal{A}^{O_{\mathsf{Sign}}}(\mathsf{pp}, \mathsf{vk}) : m^* \notin Q_{msg} \wedge \mathsf{Vf}(\mathsf{vk}, \sigma^*, m^*) = 1 \end{array}\right] ,$$

where the signing oracle O_{Sign} *takes a message* $m \in \mathcal{M}$*, runs* $\mathsf{Sign}(\mathsf{pp}, \hat{\mathsf{sgk}}, m)$ *and adds the message to a query set* Q_{msg}*. A digital signature is called* EUF-CMA*-secure if* $Adv^{\text{EUF-CMA}}_{DS,\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

The term aggregatable signature refers to those that enable the aggregation of several distinct signatures into a single one. If all the aggregated signatures are valid then the verification algorithm fulfils on the aggregated signature. Next we formally define the aggregation algorithm as a supplementary algorithm.

- $\sigma \leftarrow \mathcal{DS}.\mathsf{Agg}(\mathsf{pp}, \{\sigma_i\}_{i \in [w]})$: To aggregate w different signatures σ_i for $i \in [w]$, it takes the public parameters pp as input and then returns as aggregated signature σ as output.

BLS Signatures: Based on the formal definition of a digital signature in appendix A.1.3, we recall BLS signatures [6] as an efficient and aggregatable signature[2] For a given security parameter λ in its unary representation and cyclic group $(\mathbb{G}, \mathsf{p})$, the BLS signature consists of the following PPT algorithms:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$: Given security parameter λ, it samples $h \leftarrow\!\!\$\ \mathbb{G}$ and a hash-to-curve function $\mathsf{H} : \{0,1\}^* \rightarrow \mathbb{G}$. It returns $\mathsf{pp} = (\mathsf{H}, h)$ as output.

[2] A rogue-key attack makes aggregatable BLS signatures insecure, but the technique described in [4] addresses this problem.

- $(\mathsf{vk}, \hat{\mathsf{sgk}}) \leftarrow \mathcal{DS}.\mathsf{KGen}(\mathsf{pp})$: This algorithm takes public parameters pp and samples a random integer $s_i \leftarrow\!\!\$\ \mathbb{Z}_{\mathsf{p}}^*$ and returns the pair of signing/verification keys $(\hat{\mathsf{sgk}}_i, \mathsf{vk}_i) = (s_i, h^{s_i})$ for all $i \in [1, w]$.
- $\sigma_i \leftarrow \mathcal{DS}.\mathsf{Sign}(\mathsf{pp}, \hat{\mathsf{sgk}}_i, m)$: This probabilistic algorithm takes the pp, signing key $\hat{\mathsf{sgk}}_i$ and message $m \in \mathcal{M}$ as inputs and then computes $\sigma_i = \mathsf{H}(m)^{s_i}$ and returns the signature σ_i as output.
- $\sigma \leftarrow \mathcal{DS}.\mathsf{Agg}(\mathsf{pp}, \{\sigma_i\}_{i\in[1,w]})$: The aggregation algorithm takes pp, a set of distinct signatures $\{\sigma_i\}_{i\in[1,w]}$ and computes $\sigma = \prod_{i=1}^{w} \sigma_i$ and returns the aggregated signature σ as output.
- $0/1 \leftarrow \mathcal{DS}.\mathsf{Vf}(\mathsf{vk}, \sigma, m)$: The verification algorithm takes $\mathsf{vk} = \{\mathsf{vk}_i\}_{i=1}^{w}$, aggregated signature σ and message m as inputs. It return 1 (accept), if $\sigma \in \mathbb{G}_1$ and the equation $e(\sigma, h) = e\left(\mathsf{H}(m), \prod_{i=1}^{w} \mathsf{vk}_i\right)$ holds, otherwise it returns 0 (reject).

A digital signature is called structure-preserving [1], when it preserves the group structure over bilinear group setting, if it satisfies the following criteria:

- The verification key consists of $\mathbb{G}_1$ and $\mathbb{G}_2$ group elements.
- The signature consists of group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$.
- The messages are composed of $\mathbb{G}_1$ and $\mathbb{G}_2$ elements.
- Only $\mathbb{G}_1$ and $\mathbb{G}_2$ membership and pairing product equations of the form of $\prod_i \prod_j \hat{e}(G_i, H_j)^{c_{i,j}} = 1_T$ need to be considered in the verification algorithm, where $G_i \in \mathbb{G}_1$ and $H_j \in \mathbb{G}_2$ and $c_{i,j} \in \mathbb{Z}_{\mathsf{p}}$.

By avoiding structure-destroying operations such as hash functions, SPS are able to construct efficient schemes when combined with other primitives such as Zero-Knowledge proof systems. In a SPS, both signed messages and signatures are group elements that can be used to verify the validity of a signature by performing pairing-product equations. These unique properties make the SPS schemes attractive for a variety of privacy-preserving applications, like anonymous credentials [20, 9], anonymous e-cash [2] and access control encryptions [26]. Moreover, these signatures are efficiently re-randomizable under the knowledge of a secret randomness such that the re-randomized and original signatures are computationally indistinguishable. We utilise this property of re-randomization to ensure unlinkability of transactions in our construction.

Given the fact that the SPS relies on a single issuer then it does not meet our constructions desirable properties. In this aim, we recall the definition of Threshold

Structure-Preserving Signatures from a recent work of Crites et al. [10]: it preserves the SPS's properties while mitigating the needed trust to a single entity.

Definition 11 (Threshold Structure-Preserving Signatures [10])**.** *For a given security parameter* λ *and an asymmetric bilinear group (given in definition 4), a* (n,t)*-TSPS over message space* $\mathcal{M}$*, consists of the following PPT algorithms:*

- $\mathsf{pp} \leftarrow \mathcal{TSPS}.\mathsf{Setup}(1^\lambda)$*: Given the security parameter* λ *in its unary representation as input, it returns the set of global public parameters* pp *as the output.*
- $(\hat{\mathsf{sgk}}, \mathsf{vk}, \mathsf{vk}) \leftarrow \mathcal{TSPS}.\mathsf{KGen}(\mathsf{pp}, t, n)$*: Given the global public parameters* pp*, and integers* $t, n \in \mathsf{poly}(1^\lambda)$ *s.t.* $1 \leq t \leq n$*, as inputs it returns the vectors of secret signing keys* $\hat{\mathsf{sgk}} = (\hat{\mathsf{sgk}}_1, \ldots, \hat{\mathsf{sgk}}_n)$ *and verification keys* $\mathsf{vk} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ *along with a common verification key* vk *as output.*
- $\sigma_i \leftarrow \mathcal{TSPS}.\mathsf{ParSign}(\mathsf{pp}, \hat{\mathsf{sgk}}_i, m)$*: Given the public parameter* pp*, the* i^{th} *signing key* $\hat{\mathsf{sgk}}_i$ *and a message* $m \in \mathcal{M}$ *as inputs, it returns the partial signature* σ_i *as output.*
- $0/1 \leftarrow \mathcal{TSPS}.\mathsf{Par\text{-}Vf}(\mathsf{pp}, \mathsf{vk}_i, m, \sigma_i)$*: Given the* i^{th} *verification key* vk_i*, message* $m \in \mathcal{M}$ *and partial signature* σ_i *as inputs, it returns either* 1 *(accept) or* 0 *(reject).*
- $(\sigma, \perp) \leftarrow \mathcal{TSPS}.\mathsf{Reconst}(\mathsf{pp}, \{i, \sigma_i\}_{i \in \mathcal{T}})$*: Given* pp *and successfully verified partial signatures* $\{i, \sigma_i\}$ *over subset* $\mathcal{T} \in \{1, \ldots, n\}$ *as inputs, it returns a reconstructed signature* σ *if* $|\mathcal{T}| \geq t$*, otherwise it responds with* $\perp$*.*
- $0/1 \leftarrow \mathcal{TSPS}.\mathsf{Vf}(\mathsf{pp}, \mathsf{vk}, m, \sigma)$*: Given* pp*, verification key* vk*, a message* $m \in \mathcal{M}$ *and a reconstructed signature* σ *as inputs, it outputs either* 1 *(accept) or* 0 *(reject).*

As discussed in [10], two main security requirements for a TSPS scheme over the indexed Diffie-Hellman message spaces (as defined in definition 5) are correctness and threshold existential unforgeability under chosen indexed message attacks. We refer the readers to [10] for more details.

We recall the proposed non-interactive TSPS in [10] based on the Pedersens's Distributed Key Generation [22] as follows.

- $(\mathsf{pp}) \leftarrow \mathcal{TSPS}.\mathsf{Setup}(1^\lambda)$: It executes the bilinear pairing group generator $\mathcal{BG}(1^\lambda)$ and returns $\mathsf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h, \mathsf{p})$ as output.
- $(\hat{\mathsf{sk}}, \mathsf{vk}, \mathsf{vk}) \leftarrow \mathcal{TSPS}.\mathsf{KGen}(\mathsf{pp}, t, n)$: For a given group of authorities $\{Au_1, \ldots, Au_n\}$, it acts as follows:

1. Each authority Au_i, for $1 \leq i \leq n$, samples two initial random integers $(\alpha_{i0}, \beta_{i0}) \leftarrow\$ \mathbb{Z}_{\mathsf{p}}^*$ and does the following:
   a) It samples t random pairs $\{\alpha_{ij}, \beta_{ij}\}_{j=1}^{t}$ and calculates two polynomials $A_i[X] = \alpha_{i0} + \alpha_{i1}X + \ldots + \alpha_{it}X^t \in \mathbb{Z}_{\mathsf{p}}[X]$ and $B_i[X] = \beta_{i0} + \beta_{i1}X + \ldots + \beta_{it}X^t \in \mathbb{Z}_{\mathsf{p}}[X]$ of degree t and then commits the randomly sampled coefficients by computing, $T_{ij} = (T_{1ij}, T_{2ij}) = (h^{\alpha_{ij}}, h^{\beta_{ij}})\ \forall j \in \{0, \ldots, t\}$.
   b) Au_i transfers the pair of $(A_i(\ell), B_i(\ell))$ to Au_ℓ s.t. $\ell \in \{1, \ldots, n\} \setminus \{i\}$ and keeps $(A_i(i), B_i(i))$ secret.
2. Authority Au_i to checks the validity of the received shares from authority Au_ℓ, $(F_\ell(i), G_\ell(i))$, computes $h^{A_\ell(i)} = \prod_{j=0}^{t} T_{1\ell j}^{i^j}$ and $h^{B_\ell(i)} = \prod_{j=0}^{t} T_{2\ell j}^{i^j}$. It accepts the shares if these equations hold, else it rejects the shares and report the faulty authority Au_ℓ.
3. An authority is called disqualified if it receives at least t complaints.Lastly t qualified authorities, $Q \subset \mathcal{AU}$, continue the next steps.
4. The qualified authorities compute the global verification key as $\mathsf{vk} := (\mathsf{vk}_1, \mathsf{vk}_2) := (\prod_{i \in Q} T_{1i0}, \prod_{i \in Q} T_{2i0}) = (h^{\sum_{i \in Q} \alpha_{i0}}, h^{\sum_{i \in Q} \beta_{i0}})$.
5. Any $Au_i \in Q$ sets its private key share $\hat{\mathsf{sk}}_i$ as a pair of $\hat{\mathsf{sk}}_i = (\mathsf{sk}_{i,1}, \mathsf{sk}_{i,2}) = \left(\sum_{\ell \in Q} A_\ell(i), \sum_{\ell \in Q} B_\ell(i)\right)$.
6. Respectively the verification key vk_i can be generated by computing, $\mathsf{vk}_i = \left(h^{A(i)}, h^{B(i)}\right) = \left(\prod_{\ell \in Q} \prod_{j=0}^{t} (T_{1\ell j})^{i^j}, \prod_{\ell \in Q} \prod_{j=0}^{t} (T_{2\ell j})^{i^j}\right)$, where $A[X] = \sum_{\ell \in Q} A_\ell[X]$ and $B[X] = \sum_{\ell \in Q} B_\ell[X]$.
7. For any $Au_j \notin Q$, we have $\hat{\mathsf{sk}}_j = (0,0)$ and corresponding verification key $\mathsf{vk}_j = (1_{\mathbb{G}_2}, 1_{\mathbb{G}_2})$.

It then returns the vectors $\hat{\mathsf{sk}} = (\hat{\mathsf{sk}}_1, \ldots, \hat{\mathsf{sk}}_n)$ and $\mathsf{vk} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ for each party Au_i for $i \in [n]$ along with a common verification key vk.

- $(\sigma_i) \leftarrow \mathcal{TSPS}.\mathsf{ParSign}(\mathsf{pp}, \hat{\mathsf{sk}}_i, M)$: An authority $Au_i \in Q$ possesses $\hat{\mathsf{sk}}_i$ given an indexed DH message $M := (id, M_1, M_2) \in \mathcal{M}_{\mathsf{iDH}}^{\mathsf{H}}$ as input, it runs the hash function $\mathsf{H}(id)$ to get the random basis $\boldsymbol{g}$. If $e(\boldsymbol{g}, M_2) = e(M_1, h)$, it computes the partial signature $\sigma_i = (\boldsymbol{g}, s_i) = (\boldsymbol{g}, \boldsymbol{g}^{\mathsf{sk}_{i1}} M_1^{\mathsf{sk}_{i2}})$ and returns σ_i as output, else it responds by $\perp$.

- $0/1 \leftarrow \mathcal{TSPS}.\mathsf{Par\text{-}Vf}(\mathsf{pp}, \mathsf{vk}_i, \tilde{M}, \sigma_i)$: The partial verification algorithm takes the ith verification key, vk_i, a partial signature σ_i and message $\tilde{M} := (M_1, M_2) \in \mathcal{M}_{\mathsf{iDH}}$ as inputs. If all conditions: $M_1, s_i \in \mathbb{G}_1, \boldsymbol{g} \neq 1_{\mathbb{G}_1}$ and $M_2 \neq 1_{\mathbb{G}_2}$, $e(\boldsymbol{g}, M_2) = e(M_1, h)$ and $e(\boldsymbol{g}, \mathsf{vk}_{i1}) e(M_1, \mathsf{vk}_{i2}) = e(s_i, h)$ hold, then it returns 1 and accepts the partial signature; otherwise it returns 0 and rejects it.

- $(\sigma, \perp) \leftarrow \mathcal{TSPS}.\mathsf{Reconst}(\mathsf{pp}, \{i, \sigma_i\}_{i \in \mathcal{T}})$: Given a set of well-formed partial signatures $\{\sigma_i\}_{i \in \mathcal{T}}$, it returns a reconstructed signature by computing $\sigma :=$

$(\boldsymbol{g}, s) := \left(\boldsymbol{g}, \prod_{i \in \mathcal{T}} s_i^{L_i^{\mathcal{T}}(0)}\right)$, where $L_i^{\mathcal{T}}(0)$ is the Lagrange coefficient for the ith index corresponding to set $\mathcal{T}$ (see appendix A.1.2) and returns the aggregated signature σ as output iff $|\mathcal{T}| \geq t$, otherwise it returns $\perp$.

- $0/1 \leftarrow \mathcal{TSPS}.\mathsf{Vf}(\mathsf{pp}, \mathsf{vk}, \tilde{M}, \sigma)$: If all conditions including: $\boldsymbol{g}, M_1, s \in \mathbb{G}_1$, $\boldsymbol{g} \neq 1_{\mathbb{G}_1}$ and $M_2 \neq 1_{\mathbb{G}_2}$, $e(\boldsymbol{g}, M_2) = e(M_1, h)$ and $e(\boldsymbol{g}, \mathsf{vk}_1)e(M_1, \mathsf{vk}_2) = e(s, h)$ hold, then the verification algorithm responds by 1 (accept), else it returns 0 (reject).

The correctness and Threshold EUF-CiMA-security with adaptive adversary is formally proved in [10]. Note that in fig. 6.5 no $\mathcal{TSPS}.\mathsf{Reconst}(.)$ algorithm is operated upon, and a user may execute this algorithm if it receives at least t partial signatures from the authorities.

A.1.4 Commitment schemes

A commitment scheme [7] is a strong cryptographic primitive, which allows a committer to commit to a secret value with two main security properties, i.e., *Perfect Hiding* and *Computational Binding*. Informally, perfect hiding guarantees that the commitment does not reveal any information about the hidden committed value. Computational binding ensures that a committer cannot open a commitment under two distinct messages.

Definition 12 (Commitment schemes [7])**.** *A commitment scheme, Ψ_{CO}, over the message space of $\mathcal{M}$ and opening space of $\mathcal{T}$ consists of the following PPT algorithms:*

- $(\mathsf{pp}) \leftarrow \mathcal{CO}.\mathsf{Setup}(1^{\lambda})$*: The setup algorithm takes the security parameter λ in its unary representation as input and returns the public parameters* pp *as output.*
- $(\mathsf{com}) \leftarrow \mathcal{CO}.\mathsf{Com}(\mathsf{pp}, m)$*: The commitment algorithm takes the public parameters* pp *and a message $m \in \mathcal{M}$ as inputs, and outputs a commitment $\mathsf{com} \in \mathcal{C}$ computed under the random opening value $\tau \in \mathcal{T}$.*
- $0/1 \leftarrow \mathcal{CO}.\mathsf{Vf}(\mathsf{pp}, \mathsf{com}, m', \tau')$*: The verification algorithm is a deterministic algorithm that given commitment* com $\in \mathcal{C}$*, public parameters* pp*, a message $m' \in \mathcal{M}$ and an opening value $\tau' \in \mathcal{T}$ as inputs, returns a bit that indicates either accept (1) or reject (0).*

The primary security requirements for a commitment can be defined as follows:

Definition 13 (Correctness)**.** *A commitment scheme, Ψ_{CO}, satisfies* correctness*, if we have:*

$$\Pr\left[\forall\, m \in \mathcal{M} \;\wedge\; (\mathsf{pp}) \leftarrow \mathsf{Setup}(1^{\lambda}) : \mathsf{Vf}\left(\mathsf{pp}, m, \mathsf{Com}(\mathsf{pp}, m; \tau), \tau\right) = 1 \quad \right] \geq 1 - \mathsf{negl}(\lambda) \ .$$

Definition 14 (Computationally Hiding)**.** *A commitment, Ψ_{CO}, satisfies Computationally Hiding, if for all PPT adversaries $\mathcal{A}$ we have:*

$$\left| 2\Pr\left[\begin{array}{l}(\mathsf{pp}) \leftarrow \mathsf{Setup}(1^{\lambda}), (m_0, m_1) \leftarrow\!\$\ \mathcal{A}^{\mathsf{Com}(.)}(\mathsf{pp}), b \leftarrow\!\$\ \{0,1\}, \\ (\mathsf{com}_b) \leftarrow \mathsf{Com}(\mathsf{pp}, m_b), b' \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{com}_b) : b == b'\end{array}\right] - 1 \right| .$$

The commitment scheme is called computationally and perfectly hiding if the above probability is $\leq \mathsf{negl}(\lambda)$ and equal to 0, respectively.

Definition 15 (Computationally Binding)**.** *A cryptographic commitment scheme, Ψ_{com}, meets computationally binding security, if for all PPT adversaries $\mathcal{A}$ we have,*

$$\Pr\left[\begin{array}{l}(\mathsf{pp}) \leftarrow \mathsf{Setup}(1^{\lambda}), ((\mathsf{com}, m_0, \tau_0), (\mathsf{com}, m_1, \tau_1)) \leftarrow\!\$\ \mathcal{A}^{\mathsf{Com}(.)}(\mathsf{pp}) : \\ \mathsf{Vf}(\mathsf{pp}, \mathsf{com}, m_0, \tau_0) = \mathsf{Vf}(\mathsf{pp}, \mathsf{com}, m_1, \tau_1) = 1 \ \wedge\ m_0 \neq m_1\end{array}\right] \leq \mathsf{negl}(\lambda) \ .$$

The commitment scheme is called perfectly binding if the above probability is 0.

Our efficient instantiation utilizes Pedersen commitment [23] that is defined as follows:

- $\mathsf{pp} \leftarrow \mathcal{CO}.\mathsf{Setup}(1^{\lambda})$: It takes the security parameter, λ, as input, picks two generators $g \leftarrow\!\$\ \mathbb{G}$ and $g_1 \leftarrow\!\$\ \mathbb{G}$ uniformly at random and returns the public parameters $\mathsf{pp} = (\mathbb{G}, \mathsf{p}, g, g_1)$ as output.
- $\mathsf{com} \leftarrow \mathcal{CO}.\mathsf{Com}(\mathsf{pp}, m)$: It takes the public parameters pp and a message $m \in \mathbb{Z}_{\mathsf{p}}$ as inputs, picks random opening $\tau \leftarrow\!\$\ \mathbb{Z}_{\mathsf{p}}^{*}$ and computes and outputs $\mathsf{com} = g^{\tau} g_1^{m}$.
- $0/1 \leftarrow \mathcal{CO}.\mathsf{Vf}(\mathsf{pp}, \mathsf{com}, m', \tau')$: It computes $\mathsf{com}' = g^{\tau'} g_1^{m'}$, if $\mathsf{com} = \mathsf{com}'$ it accepts and returns 1: else it responds with 0 and rejects the commitment.

A.1.5 Threshold Encryption

Next, recall the definition of threshold encryption schemes and then slightly modify it and define the notion of randomness reusable threshold encryption schemes.

Definition 16 (Threshold Encryption)**.** *Given a set of public parameters* pp *along with security parameter λ, a (n,t)-threshold encryption, Ψ_{TE}, over the message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$ consists of four main PPT algorithms defined as follows:*

- $(\hat{\mathsf{sk}}, \mathsf{pk}) \leftarrow \mathcal{TE}.\mathsf{KGen}(\mathsf{pp}, n, t)$*: Key generation is a probabilistic algorithm that takes the set of public parameter* pp *along with two integers $t, n \in \mathsf{poly}(\lambda)$ as inputs. It then returns a vector of secret key* $\hat{\mathsf{sk}}$ *of size n and a general public key* pk *as outputs.*

- $\mathtt{Ct} \leftarrow \mathcal{TE}.\mathsf{Enc}(\mathsf{pp},\mathsf{pk},m)$*: The encryption algorithm as a probabilistic algorithm takes the public parameters* pp*, global public key* pk *and a message* $m \in \mathcal{M}$ *as inputs. It returns ciphertext* $\mathtt{Ct} \in \mathcal{C}$ *as output. When we want to assign a specific value to the random integer r, we write* $\mathsf{Enc}(\mathsf{pp},\mathsf{pk},m;r)$.

- $\mathsf{pd}_j \leftarrow \mathcal{TE}.\mathsf{PDec}(\mathsf{pp},\mathsf{sk}_j,\mathtt{Ct})$*: The partial decryption algorithm is run by receiver* $j \in \mathcal{R}$ *and takes the public parameters* pp*, the secret key* $\hat{sk}_j$ *of the receiver j and a ciphertext* $\mathtt{Ct}$ *as inputs. It returns a partially decrypted ciphertext* pd_j *as output.*

- $(\perp,m) \leftarrow \mathcal{TE}.\mathsf{Dec}(\mathsf{pp},\mathtt{Ct},\{\mathsf{pd}_j\}_{j\in\mathcal{K}})$*: The decryption algorithm takes the public parameters* pp*, a ciphertext* $\mathtt{Ct}$ *and the partially decrypted ciphertexts* $\{\mathsf{pd}_j\}_{j\in\mathcal{K}}$ *as inputs. If* $|\mathcal{K}| \geq t$*, it returns* $m \in \mathcal{M}$*, else it responds by* $\perp$.

The primary security requirements for a (n,t)-threshold encryption are correctness and *static semantic security* and *partial decryption simulatability* based on the static security definitions of Reyzen et al. [24].

Definition 17 (Correctness)**.** *A* (n,t)*-threshold encryption,* Ψ_{TE}*, for all security parameters* λ *and messages* $m \in \mathcal{M}$ *and set of receivers* $\mathcal{R}$ *is called correct if for any* $|\mathcal{K}| \geq t$ *we have:*

$$\Pr\left[\begin{array}{l}\forall \mathsf{pp},\lambda,n,t,(\hat{\mathsf{sk}},\mathsf{pk}) \leftarrow \mathsf{KGen}(\mathsf{pp},n,t): \\ \\ \mathsf{Dec}\left(\mathsf{pp},\mathsf{Enc}(\mathsf{pp},\mathsf{pk},m),\{\mathsf{PDec}(\mathsf{pp},\hat{\mathsf{sk}}_j,\mathtt{Ct})\}_{j\in\mathcal{K}}\right)=m\end{array}\right] \geq 1-\mathsf{negl}(\lambda) \ .$$

Definition 18 (Static Semantic Security [24])**.** *A* (n,t)*-threshold encryption,* Ψ_{TE}*, is said to be* (n,t)*-statically semantic secure (SSS) if for all PPT adversaries* $\mathcal{A}$ *in winning the following experiment we have* $\Pr[\mathrm{EXP}_{\mathcal{A}}^{SSS}(1^{\lambda},\mathcal{R},t)=1] \leq 1/2-\mathsf{negl}(\lambda)$*. Where adversary* $\mathcal{A}$ *has access to a partial decryption oracle,* O_{PDec}*, and can obtain up to* $t-1$ *partially decrypted values of the given ciphertext.*

$\text{EXP}^{SSS}_{\mathcal{A}}(1^\lambda, \mathcal{R}, t)$

1 : $\mathcal{C} \leftarrow \mathcal{A}(1^\lambda, \mathcal{R}, t)$

2 : $(\hat{\mathsf{sk}}_i) \leftarrow \mathcal{TE}.\mathsf{KGen}(\mathsf{msk}, i)$ For $i \in [n]$;

3 : $(m_0, m_1) \leftarrow \mathcal{A}^{O_{\text{PDec}}}(\{\hat{\mathsf{sk}}_i\}_{i \in \mathcal{C}})$;

4 : $b \leftarrow\$ \{0,1\}$,

5 : $(\mathtt{Ct}_b) \leftarrow \mathcal{TE}.\mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, m_b)$;

6 : $b' \leftarrow\$ \mathcal{A}^{O_{\text{PDec}}}(\mathtt{Ct}_b)$;

7 : **return** $(b' = b \wedge |m_0| \neq |m_1| \wedge |\mathcal{R} \cap \mathcal{K}| < t)$

Figure A.2: Static Semantic Security Experiment

A.1.6 Collaborative Key Generation and RRTE

Collaborative Key Generation (CKG) is a slightly modified variant of DKG that uses threshold cryptography to achieve distributed key generation. The (n,t)-DKG generates n pairs of secret and public keys s.t. at least t parties among n are required to execute a key oriented operation, whereas any subset of size smaller than t is not able to execute it. Likewise in a (n,t,k)-CKG, a global public key pk corresponding to a secret key sk is shared among n parties s.t. any subset of larger than k can rebuild pk and any subset of larger than t, where $k < t \leq n$ can reconstruct the sk. The main difference here is that any operation that needs the secret key sk requires the cooperation of at least t collaborators, whereas the public key pk itself can be reconstructed by at least $k < t$ parties. For a given public parameters $\mathsf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, h, \mathsf{p}, e)$, we instantiate a CKG construction as follows:

- $(\mathsf{pk}, \mathsf{pk}) \leftarrow \mathsf{CKG}(\mathsf{pp}, k, t, k)$: The CKG is a collaborative algorithm that is executed by the group of authorities $\mathcal{AU}$ of size n to generate public keys for ℓ merchants, defined in the fig. 6.5 with a fixed threshold $k = 2$, while their corresponding secret keys remain hidden as long as the majority of the authorities are honest ($t \geq n/2 + 1$).

    1. Each authority $\mathcal{AU}_i$ samples an initial random value $x_{i0} \leftarrow\$ \mathbb{Z}^*_{\mathsf{p}}$ and does the following:

a) It samples a random integer $\{x_{i1}\}$, forms a polynomial $F_i[X] = x_{i0} + x_{i1}X \in \mathbb{Z}_{\mathsf{p}}[X]$ and commits the coefficients by publishing, $V_{ij} = h^{x_{ij}}\ \forall\ j \in \{0,1\}$.

b) It broadcasts $F_i(j)$ to the rest of authorities as a share corresponding to the jth merchants.

2. Each authority checks the consistency of the received shares, $F_i(j)$, from $\mathcal{AU}_i$ by computing the equations $g^{F_i(j)} = V_{i0}V_{i1}^{j}$ for all merchants' label $j \in [\ell]$. If this equation holds, the shares generated by $\mathcal{AU}_i$ will be accepted, otherwise it will reject and then report the faulty authority $\mathcal{AU}_i$.
3. Any faulty authority that receives at least $t \geq n/2+1$ complaints is labelled as disqualified. At the end of this phase t parties from the set of qualified authorities, $Q \subset \mathcal{AU}$ perform the next steps.
4. The global public key is determined as $\mathsf{pk} := \prod_{i\in Q} V_{i0} = h^{\sum_{i\in Q} x_{i0}}$.
5. Each merchant $\mathcal{M}_j$ is assigned by a public key pk_j that is obtained by computing, $\mathsf{pk}_j = h^{F(j)} = \prod_{i\in Q}\left(V_{i0}(V_{i1})^{j}\right)$, where $F[X] = \sum_{i\in Q} F_i[X]$.

These steps complete the CKG phase and return the set of public keys $\{\mathsf{pk}_j\}_{1\leq j\leq \ell}$ along with the global public key pk.

An efficient RRTE. We propose an efficient RRTE scheme that relies on CKG constructions appendix A.1.6.

- $(\mathsf{pk}, \mathsf{pk}) \leftarrow \mathcal{RRTE}.\mathsf{KGen}(\mathsf{pp}, \ell, t, 2)$: It runs the CKG algorithm, $\mathsf{CKG}(\mathsf{pp}, \ell, k, t, 2)$, as described in appendix A.1.6.
- $(\mathtt{Ct}_j, v) \leftarrow \mathcal{RRTE}.\mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, m, \mathsf{pk}_j)$: The encryption algorithm takes public parameters pp, global public key pk, the message m and public key pk_j as inputs. It samples $r \leftarrow\$\ \mathbb{G}_1$ uniformly at random and generates the ciphertext underlying each recipient by computing $(\mathtt{Ct}_j, v) := \left(e(r, \mathsf{pk}_j), m \cdot e(r, \mathsf{pk})\right)$.
- $(m, \perp) \leftarrow \mathcal{RRTE}.\mathsf{Dec}(\mathsf{pp}, \mathtt{Ct}_j, \mathtt{Ct}_{j'}, v)$: The decryption algorithm takes twin ciphertexts $(\mathtt{Ct}_j, v)$ and $(\mathtt{Ct}_{j'}, v)$ along with public parameters pp as inputs. Let $\mathcal{J} = \{j, j'\}$, it computes $g_T^{rs} : z = \left(\mathtt{Ct}_j^{L_j^{\mathcal{J}}(0)} \cdot \mathtt{Ct}_{j'}^{L_{j'}^{\mathcal{J}}(0)}\right)$ and then returns $m := v/z$, otherwise, it responds with $\perp$.

A.1.7 Non-Interactive Zero-Knowledge proofs

Zero-Knowledge proofs [18] are two-party protocols, which are a fundamental and powerful cryptographic tool. They allow a prover to convince the verifier about the

validity of a statement without revealing any other information. Non-Interactive Zero-Knowledge proofs remove the interaction between the parties in two possible settings, either the Random Oracle Model (ROM) [15] or the Common Reference String (CRS) model [3]. We recall the definition of NIZK arguments[3] in the CRS model, in which the prover is computationally bounded to ensure the soundness. Hence, for security parameter λ, let $\mathcal{R}$ be a relation generator, such that $\mathcal{R}(1^\lambda)$ returns an efficiently computable binary relation $\mathbf{R_L} = \{(x, w)\}$, where x is the instance (statement) and w is the corresponding witness. Let $\mathbf{L} = \{x : \exists\, w \mid (x, \hat{w}) \in \mathbf{R_L}\}$ be the NP-language consisting of the statements in relation $\mathbf{R_L}$.

Definition 19 (NIZK arguments)**.** *Formally, a NIZK argument,* Ψ_{NIZK}*, for* $\mathcal{R}$ *consists of a tuple of PPT algorithms* $\mathcal{ZK}.(\mathsf{K_{crs}}, \mathsf{P}, \mathsf{Vf}, \mathsf{Sim})$*, defined as follows:*

- $(\mathsf{crs}, \hat{\mathsf{ts}}, \hat{\mathsf{te}}) \leftarrow \mathcal{ZK}.\mathsf{K_{crs}}(1^\lambda, \mathbf{R_L})$*: The CRS generator as a probabilistic algorithm takes the security parameter* 1^λ *and relation* $\mathbf{R_L}$ *as inputs. It then generates common reference string* crs *by sampling a simulation trapdoor* $\hat{\mathsf{ts}}$ *and an extraction trapdoor* $\hat{\mathsf{te}}$*. It keeps the trapdoors* $(\hat{\mathsf{te}}, \hat{\mathsf{ts}})$ *hidden while publishes* crs*.*
- $(\pi, \perp) \leftarrow \mathcal{ZK}.\mathsf{P}(\mathbf{R_L}, \mathsf{crs}, x, \hat{w})$*: Prove as a probabilistic algorithm takes the CRS,* crs*, and a pair of statement and witness* $(x, \hat{w})$ *as inputs. If* $(x, \hat{w}) \in \mathbf{R_L}$ *it returns a proof* π*, otherwise it responds with* $\perp$*. This algorithm sometimes is denoted by* $\mathsf{PoK}\{\hat{w} \mid (x, \hat{w}) \in \mathbf{R_L}\}$*.*
- $0/1 \leftarrow \mathcal{ZK}.\mathsf{Vf}(\mathbf{R_L}, \mathsf{crs}, x, \pi)$*: Verification as a deterministic algorithm takes CRS,* crs*, and a pair of statement and proof* (x, π) *as inputs. It either returns* 1 *(accept) or* 0 *(reject).*
- $\pi' \leftarrow \mathcal{ZK}.\mathsf{Sim}(\mathbf{R_L}, \mathsf{crs}, \hat{\mathsf{ts}}, x)$*: The Simulator algorithm takes the tuple* $(\mathbf{R_L}, \mathsf{crs}, \mathsf{ts}, x)$ *as input and without knowing the corresponding secret witness, outputs a simulated proof* π' *s.t. it is computationally indistinguishable from* π*.*

Definition 20 (Completeness)**.** *A NIZK argument,* Ψ_{NIZK}*, is called* complete *for relation* $\mathbf{R_L} \in \mathcal{R}$*, if for all security parameters* 1^λ *and* $(x, \hat{w}) \in \mathbf{R_L}$*, we have:*

$$\Pr\left[(\mathsf{crs}, \hat{\mathsf{ts}}, \hat{\mathsf{te}}) \leftarrow \mathsf{K_{crs}}(1^\lambda, \mathbf{R_L}) : \mathsf{Vf}(\mathbf{R_L}, \mathsf{crs}, x, \mathsf{P}(\mathbf{R_L}, \mathsf{crs}, x, \hat{w})) = 1 \quad \right] \geq 1 - \mathsf{negl}(\lambda) \ .$$

Definition 21 (Soundness)**.** *A NIZK argument,* Ψ_{NIZK}*, is* Sound *for any relation* $\mathbf{R_L} \in \mathcal{R}$*, if for all PPT adversaries* $\mathcal{A}$*, we have:*

$$\Pr\left[(\mathsf{crs}, \hat{\mathsf{ts}}, \hat{\mathsf{te}}) \leftarrow \mathsf{K_{crs}}(1^\lambda, \mathbf{R_L}), (x, \pi) \leftarrow \mathcal{A}(\mathbf{R_L}, \mathsf{crs}) : \mathsf{Vf}(\mathbf{R_L}, \mathsf{crs}, x, \pi) = 1 \wedge x \notin \mathbf{L_R} \quad \right] \leq \mathsf{negl}(\lambda) \ .$$

Definition 22 (Statistically Zero-Knowledge)**.** *A NIZK argument,* Ψ_{NIZK}*, is called* statistically Zero-Knowledge*, if for all security parameter* 1^λ*, and all PPT adversaries*

[3]The CRS does not depend on the language distribution or language parameters.

$\mathcal{A}$ we have, $\varepsilon_0^{unb} \approx_\lambda \varepsilon_1^{unb}$, where,

$$\varepsilon_b = \Pr\left[(\mathsf{crs}, \hat{\mathsf{ts}}, \hat{\mathsf{te}}) \leftarrow \mathsf{K}_{\mathsf{crs}}(1^\lambda, \mathbf{R_L}) : \mathcal{A}^{\mathsf{O}_b(\cdot,\cdot)}(\mathbf{R_L}, \mathsf{crs}) = 1\right].$$

Here, the oracle $\mathcal{O}_0(\mathsf{x}, \hat{\mathsf{w}})$ returns $\perp$ (reject) if $(\mathsf{x}, \hat{\mathsf{w}}) \notin \mathbf{R_L}$, otherwise it returns $\mathcal{ZK}.\mathsf{P}(\mathbf{R_L}, \mathsf{crs}, \mathsf{x}, \hat{\mathsf{w}})$. Similarly, $\mathcal{O}_1(\mathsf{x}, \hat{\mathsf{w}})$ returns $\perp$ (reject) if $(\mathsf{x}, \hat{\mathsf{w}}) \notin \mathbf{R_L}$, otherwise it returns $\mathcal{ZK}.\mathsf{Sim}(\mathbf{R_L}, \mathsf{crs}, \mathsf{x}, \hat{\mathsf{ts}})$.

Definition 23 (Computational Knowledge-Soundness)**.** *A NIZK argument, Ψ_{NIZK}, is called computationally (adaptively)* knowledge-sound for *$\mathcal{R}$, if for all PPT adversary $\mathcal{A}$ and $\mathbf{R_L} \in \mathcal{R}$, there exists an extractor $\mathsf{Ext}_{\mathcal{A}}$, s.t. for all 1^λ we have,*

$$\Pr\left[\begin{array}{l}(\mathsf{crs}, \hat{\mathsf{ts}}, \hat{\mathsf{te}}) \leftarrow \mathsf{K}_{\mathsf{crs}}(1^\lambda, \mathbf{R_L}), (\mathsf{x}, \pi) \leftarrow \mathcal{A}(\mathbf{R_L}, \mathsf{crs}), \\ (\hat{\mathsf{w}}) \leftarrow \mathsf{Ext}_{\mathcal{A}}(\mathbf{R_L}, \mathsf{crs}, \hat{\mathsf{te}}, \pi) : (\mathsf{x}, \hat{\mathsf{w}}) \notin \mathbf{R_L} \ \wedge \ \mathsf{Vf}(\mathbf{R_L}, \mathsf{crs}, \mathsf{x}, \pi) = 1\end{array}\right] \leq \mathsf{negl}(\lambda) \ .$$

Next, we overview the used proof systems in the implementation of the proposed efficient instantiation. As we already discussed in section 4, we rely on three main proof systems including the Sigma protocols, Groth-Sahai proof systems and Range-proofs.

Sigma Protocols [25]. In what follows we recall the standard Sigma protocols and also some techniques used recently in [11, 21]. All the protocols are interactive and we use the Fiat-Shamir paradigm to make make then non-interactive; additionally, the existence of a collision-resistance hash function $\mathsf{H} : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ modeled in random oracle is assumed. Thus, the prover runs the hash function to obtain the challenge c instead of taking the random challenges from the verifier over an extra round of communication.

Proof of knowledge of DLog. fig. A.3 outlines an interactive Sigma protocol to prove the knowledge of discrete logarithm, $x \in \mathbb{Z}_p$, of a public value g^x, where g is the generator of a cyclic group $\mathbb{G}$ with prime order p.

To prove a multiplicative relation between two committed values. fig. A.4 describes an interactive Sigma protocol to prove the knowledge of three integers x_1, x_2, x_3 with public Pedersen commitments of them, i.e. $\mathsf{com}_i = g^{\tau_i} g_1^{x_i}$ for $i = 1, 2, 3$ and also showing the fact $x_3 = x_1 x_2 \mod p$.

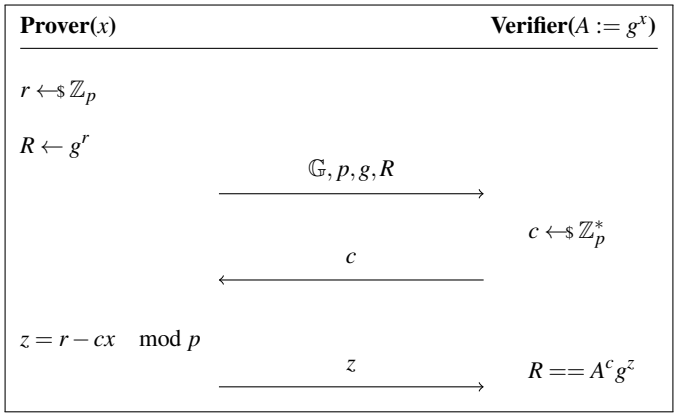


Figure A.3: Proving the knowledge of DLog

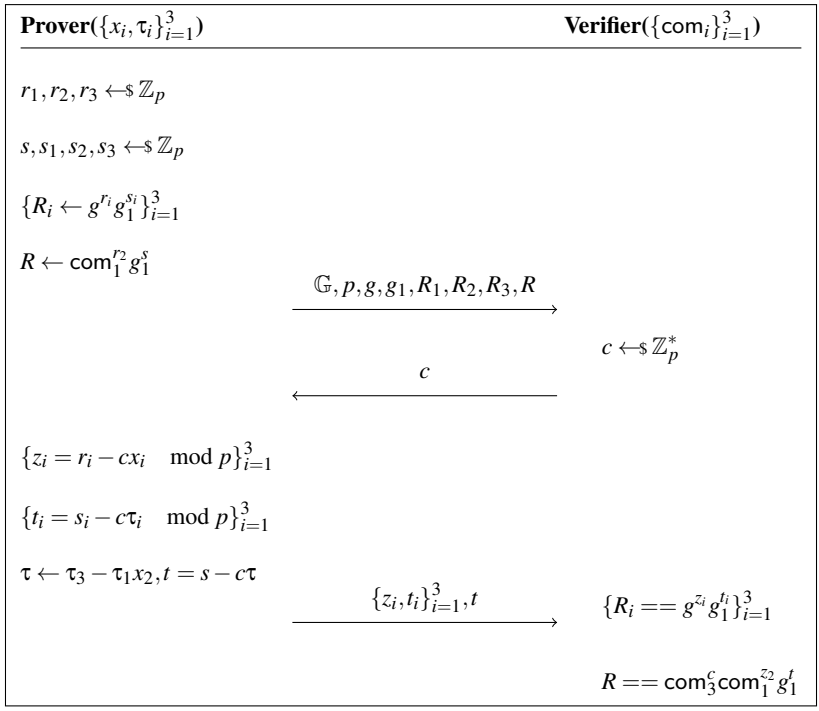


Figure A.4: To prove a multiplication relation between committed values

To prove the well-formedness of the Dodis-Yampolskiy PRF. fig. A.5 outlines a Sigma

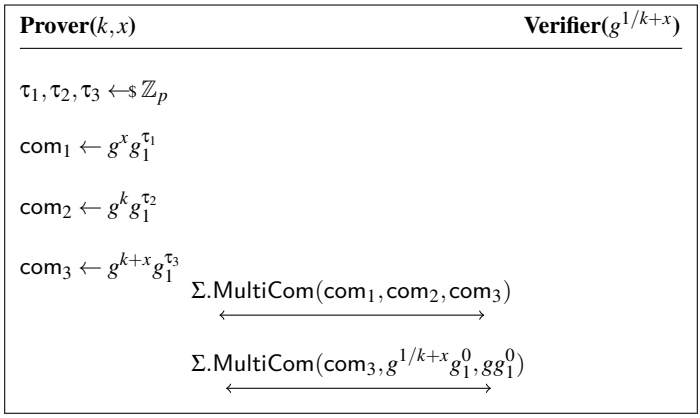


Figure A.5: Proving well-formedness of the DY's PRF

protocol to prove the knowledge of a seed $k \in \mathcal{K}_{\mathsf{PRF}}$ and showing the fact that it is correctly computed. For this, we rely on the above mentioned Sigma protocol and denote it by $\Sigma.\mathsf{MultiCom}(.)$.

Groth-Sahai (GS) proof system [19]. The GS proof system is a commit-and-prove system where the prover takes CRS and commits to the hidden variables (witnesses), subsequently providing proofs to convince the verifier that the PPE relation is valid under the generated commitments. In a subsequent work, Escala and Groth in [14] show that by replacing the commitment with ElGamal encryption [13] the prover's computation cost can be reduced. Thus in this paper the prover to commit to hidden values (witnesses) computes an ElGamal encryption of the witnesses. In the implementation, we take one of the instantiation proposed in [19] that relies on the symmetric external Diffie-Hellman (SXDH) assumption over prime order groups. Over an asymmetric bilinear group $(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,\mathsf{p},e)$, the relation over variables $\mathcal{X}_1,\ldots,\mathcal{X}_m \in \mathbb{G}_1$, $\mathcal{Y}_1,\ldots,\mathcal{Y}_n \in \mathbb{G}_2$ and a constant $T \in \mathbb{G}_T$ can be any product-pairing equation (PPE) on the form:

$$\prod_{i=1}^{n} e(A_i,\mathcal{Y}_i) \prod_{i=1}^{m} e(\mathcal{X}_i,B_i) \prod_{j=1}^{m}\prod_{i=1}^{n} e(\mathcal{X}_j,\mathcal{Y}_i)^{\mu_{i,j}} = T \ ,$$

where $(A_1,\ldots,A_n) \in \mathbb{G}_1$, $(B_1,\ldots,B_m) \in \mathbb{G}_2$ and $\{\mu_{i,j}\}_{1\le i\le m \ \&\ 1\le j\le n} \in \mathbb{Z}_{\mathsf{p}}^{m\cdot n}$ are constant.

Range-proofs. These constructions enable a prover to prove that a hidden witness x lies within a certain interval like $(0,Max]$, where Max can be any integer upper bound in this relation. In the implementation of our construction we rely on Bulletproof proposed by Bunz et al. [8].

A.1.8 NIZK Language Realization

Next, we briefly discuss how the described languages in section 3.1 are implemented.

- Language $\mathbf{L}_1$: Towards the realization of this language we rely on Sigma protocols. The customer runs the described Sigma protocol in fig. A.3 on the proof of knowledge of $\hat{\mathsf{sk}}_{cm}$.
- Language $\mathbf{L}_2$: To realize this language, we rely on Sigma protocols, Range-proofs and GS proof systems. To show the commitment k'_j is formed correctly and it is indeed the Pedersen commitment of the hidden key $\hat{k}_j$, the prover runs the described protocol in fig. A.4 on a single commitment. To be more precise, the customer can assign $x_2 = 0$ and perform this protocol. Additionally, the customer runs the range-proof on the committed PRF in the previous relation and depicts the fact that it samples correctly from the key distribution of the PRF, i.e. $\mathcal{K}_{\mathsf{PRF}}$. To show the customer has the knowledge of a valid certificate $\hat{\mathsf{cert}}_{cn}$ signed by at least t authorities, we rely on GS proof systems. As we already discussed, the verification equation of the given TSPS can be written as a pairing-product equation and the customer can convince the merchant on this. The main advantage of this proof system is that the prover does not need to necessarily know the discrete logarithm of the message and it is compatible with all group element witnesses. Moreover, this enables us to have a straight-line extraction of the witnesses and allows proof aggregation.
- Language $\mathbf{L}_3$: The instantiation of this language is done based on two proof systems: Sigma protocols and GS proof systems. To prove the well-formedness of the PRF evaluation function and showing the fact that the prover knows the key $\hat{k}_j$, the customer runs the described Sigma protocol in fig. A.5. Since the remaining relations of this language can be written as a PPE, the prover can utilise the GS proof systems.

In all of the above cases, all relations and languages are proven solely and a bridge can be made between them whenever the issued commitments are all committing to the same hidden witness. Sigma protocols can be used to demonstrate that the committed values in two distinct commitments are the same.

Bibliography

[1] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, Heidelberg, August 2010.

[2] Olivier Blazy, Sébastien Canard, Georg Fuchsbauer, Aline Gouget, Hervé Sibert, and Jacques Traoré. Achieving optimal anonymity in transferable e-cash with a judge. In Abderrahmane Nitaj and David Pointcheval, editors, *AFRICACRYPT 11*, volume 6737 of *LNCS*, pages 206–223. Springer, Heidelberg, July 2011.

[3] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 103–112. ACM, 1988.

[4] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 435–464. Springer, Heidelberg, December 2018.

[5] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.

[6] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004.

[7] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[8] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.

[9] Jan Camenisch, Maria Dubovitskaya, Kristiyan Haralambiev, and Markulf Kohlweiss. Composable and modular anonymous credentials: Definitions and practical constructions. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 262–288. Springer, Heidelberg, November / December 2015.

[10] Elizabeth Crites, Markulf Kohlweiss, Bart Preneel, Mahdi Sedaghat, and Daniel Slamanig. Threshold structure-preserving signatures. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023*, pages 348–382, Singapore, 2023. Springer Nature Singapore.

[11] Ivan Damgård, Chaya Ganesh, Hamidreza Khoshakhlagh, Claudio Orlandi, and Luisa Siniscalchi. Balancing privacy and accountability in blockchain identity management. LNCS, pages 552–576. Springer, Heidelberg, 2021.

[12] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In Serge Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 416–431. Springer, Heidelberg, January 2005.

[13] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, Heidelberg, August 1984.

[14] Alex Escala and Jens Groth. Fine-tuning Groth-Sahai proofs. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 630–649. Springer, Heidelberg, March 2014.

[15] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

[16] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008. Applications of Algebra to Cryptography.

[17] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 1984.

[18] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[19] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

[20] Lucjan Hanzlik and Daniel Slamanig. With a little help from my friends: Constructing practical anonymous credentials. pages 2004–2023. ACM Press, 2021.

[21] Ueli M. Maurer. Unifying zero-knowledge proofs of knowledge. In Bart Preneel, editor, *AFRICACRYPT 09*, volume 5580 of *LNCS*, pages 272–286. Springer, Heidelberg, June 2009.

[22] Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract) (rump session). In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 522–526. Springer, Heidelberg, April 1991.

[23] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.

[24] Leonid Reyzin, Adam Smith, and Sophia Yakoubov. Turning HATE into LOVE: Homomorphic ad hoc threshold encryption for scalable MPC. Cryptology ePrint Archive, Report 2018/997, 2018. https://eprint.iacr.org/2018/997.

[25] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, August 1990.

[26] Mahdi Sedaghat and Bart Preneel. Cross-domain attribute-based access control encryption. LNCS, pages 3–23. Springer, Heidelberg, 2021.

[27] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

Curriculum vitae

Akash Madhusudan obtained a B.Tech in Computer Science and Engineering in 2015 from SRM University, India. From 2015-2017 he worked as a Software Engineer at Cognizant, moving to Belgium in 2016 to work at KBC Bank through Cognizant Belgium.

In 2018, he earned an Advanced Master's in Artificial Intelligence from KU Leuven. He embarked on his doctoral journey at COSIC in December 2018, under the joint supervision of Prof. Dr. ir. Bart Preneel and Dr. Svetla Petkova-Nikova. Since November 2022, he has been dedicating his time to full-time research at 3MI Labs, a Belgian research house specializing in Cryptography and Distributed Systems.



FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
COSIC
Kasteelpark Arenberg 10, bus 2452
B-3001 Leuven
akash.madhusudan@esat.kuleuven.be
https://www.esat.kuleuven.be/cosic/