KATHOLIEKE UNIVERSITEIT
LEUVEN

Arenberg Doctoral School of Science, Engineering & Technology
Faculty of Engineering
Department of Electrical Engineering (ESAT)

Mathematical aspects of symmetric-key cryptography

Elmar Wolfgang TISCHHAUSER

Dissertation presented in partial
fulfilment of the requirements for
the degree of Doctor
in Engineering

May 2012

Mathematical aspects of symmetric-key cryptography

Elmar Wolfgang TISCHHAUSER

Jury:
Prof. dr. Ludo Froyen, chair
Prof. dr. ir. Vincent Rijmen, promotor
Dr. Andrey Bogdanov
Prof. dr. ir. Ronald Cools
Prof. dr. ir. Bart Preneel
Dr. Joan Daemen
(ST Microelectronics)

Dissertation presented in partial
fulfilment of the requirements for
the degree of Doctor
in Engineering

May 2012

© Katholieke Universiteit Leuven – Faculty of Engineering
Kasteelpark Arenberg 10, bus 2446, B-3001 Heverlee (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotocopie, 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 or any other means without written permission from the publisher.

D/2012/7515/51
ISBN 978-94-6018-515-1

Acknowledgements

It is a real pleasure for me to thank the many people who have helped me realise this Ph.D. and have made me feel being in the right place during the past four years.

I could not possibly begin with anyone else than my promotor, Prof. Vincent Rijmen. From the very first contact till the busy final phase, Vincent has been a driving and inspiring force for me. Among many other things, I am indebted to Vincent for always being there, for his wise advice, and not the least for dealing so gracefully with my very un-German attitude towards deadlines. The best description for his guidance is that he has been a “Doktorvater” in the best sense of the word.

I am very grateful to Dr. Andrey Bogdanov, Prof. Ronald Cools, Dr. Joan Daemen and Prof. Bart Preneel for serving in my jury, and to Prof. Carlo Vandecasteele and Prof. Ludo Froyen for chairing it. A special thanks goes to Bart for his work on supporting all my scholarship applications, and for his reliable and always fast advice. The support of the F.W.O. (Fund for Scientific Research—Flanders) is greatly appreciated.

At COSIC and beyond, I had the fortune of meeting or getting into contact with many very bright and highly knowledgeable researchers, of whom I’d particularly like to acknowledge my co-authors: Elena Andreeva, Paulo Barreto, Andrey Bogdanov, Christophe De Cannière, Orr Dunkelman, Kota Ideguchi, Sebastiaan Indesteege, Emilia Käsper, Lars Knudsen, Gregor Leander, Svetla Nikova, Venitzslav Nikov, Andreas Pashalidis, Bart Preneel, Vincent Rijmen, Stefan Schiffner, François-Xavier Standaert, John Steinberger, Yue Sun and Meiqin Wang. I am indebted to all of you.

I would like to thank Andrey for his sharp mind as well as for his sense of humour, which made our many discussions both enlightening and exhilarating. And the late-night “freak sessions” will definitely be remembered for a long time!

I want to thank Sebastiaan especially for being a “guide” after arriving in Leuven, for getting orientated in COSIC, Belgium, and in crypto. The same big thanks go to my first office mates Özgül and Christophe.

Since their arrival, Deniz and Kerem, Filipe and also Florian have been both colleagues and friends. Thank you for lots of both scientific and non-scientific, serious and funny discussions and chats!

Vesselin, I want to thank you especially for the great piano & guitar jam sessions we had, and for introducing me to Bulgarian music.

I would like to thank both Stefan and Roel (P.) for the friendship and the many interesting excursions – into the world of fine food, cultural events, and for being so patient while I was still struggling with my Dutch. I am especially indebted for your help after I broke my ankle.

If Vincent has been my “Doktorvater”, Péla certainly deserves that the word “Doktormutter” be invented for her. Thanks a lot for being so kind, for proactively helping with issues that I wasn’t even aware of yet, and let us not forget the delicious eggs!

A very deep thank you goes to my family: to my parents for their support and encouragement, to my mother in particular for coping with being the only humanities scholar in the family and still proofreading my manuscript with eagle eyes; to my sister Gundula and her husband Suleiman not the least for their hospitality that allowed us to frequently convert their home into a base camp for the Swiss mountains.

The same way I could not have begun with anyone else than Vincent, I could not save anyone else for last but her: Li, it is impossible to adequately thank you for your love, for every moment with you, and for managing to combine such a sharp mind with being so practical at the same time. It is to you who this thesis is dedicated.

Elmar Tischhauser
Leuven, May 2012

Abstract

It is hard to overestimate the ubiquity and importance of secure communications and information processing in modern society. From private individuals to industry or governments — they all rely on technology guaranteeing the confidentiality, integrity and authenticity of their communication. To realise these security goals, one relies on *cryptographic algorithms*, often totally transparent to their users.

For a cryptographic algorithm to be useful, it is important to have a good understanding to which extent it actually achieves the intended security goals. Progress in understanding how to analyse ("break") cryptographic algorithms is going to improve our understanding of how to design them, and vice versa. At the same time, obtaining rigorous statements about the security guarantees offered by algorithms on the one hand, and the power of attacks on the other hand is an important complement to the perpetual interplay between cryptanalysis and design.

This thesis is dedicated to the study of symmetric-key cryptographic algorithms, which form the backbone of virtually all security systems. It aims at improving the understanding of these algorithms by extending the mathematical foundations regarding design, analysis, and security proofs of symmetric algorithms.

As a first contribution, we propose *nonsmooth cryptanalysis*, a novel technique for the analysis of symmetric algorithms based on the application of methods from nonsmooth optimisation to the solving of equations over finite fields of characteristic two. We then focus on *rebound attacks*, a powerful recent method for the cryptanalysis of hash functions. In this context, we demonstrate new extensions of this attack on the hash function Grøstl-0, and analyse how to design a hash function which is resistant to rebound attacks, leading to the new hash function *Whirlwind*.

Incorporating the advances in cryptanalysis into new design criteria also requires a rigorous understanding of the exact power of these attacks. We study two important cryptanalysis methods, linear cryptanalysis and differential attacks using structures, and obtain more precise and realistic mathematical models for the complexity analysis of these attacks. In both cases, we base our study on a deepened analysis of the statistical phenomena exploited by these attacks.

Complementing this analysis, we consider the question of ideal statistical behaviour with regard to linear and differential cryptanalysis and some of their extensions, providing an explicit characterisation of a *reference point* for resistance against these attacks.

Finally, we study *key-alternating ciphers* from a structural point of view. With the ubiquitous Advanced Encryption Standard (AES) belonging to this class, key-alternating ciphers are a particularly important way of constructing a block cipher. We prove that key-alternating ciphers can be considered a sound construction principle. In the context of its resistance to linear attacks, our study especially highlights the constructive effect of having multiple, but not too many rounds in such a design.

Beknopte samenvatting

Het belang van veilige communicatie en informatieverwerking voor de moderne maatschappij is moeilijk te overschatten. Particulieren, bedrijven, overheden – vrijwel iedereen vertrouwt op beveiligingstechnologie om de vertrouwelijkheid, integriteit en authenticiteit van hun communicatie te waarborgen. Voor het realiseren van deze doelen worden *cryptografische algoritmes* gebruikt, vaak zonder dat de gebruikers zich dat realiseren.

Opdat een cryptografisch algoritme zijn doelstelling goed vervult, is het cruciaal om te kunnen bepalen in hoeverre het algoritme aan het gewenste veiligheidsniveau voldoet. Elke vooruitgang in de analysemogelijkheden voor cryptografische algoritmes gaat ons begrip van het ontwerpen van deze algoritmes verbeteren, en vice versa. Tegelijkertijd is het bepalen van rigoureuze bewijzen voor de veiligheidsgaranties van algoritmes enerzijds en de kracht van aanvallen anderzijds een belangrijke aanvulling op de voortdurende wisselwerking tussen cryptanalyse en ontwerp van cryptografische algoritmes.

Dit proefschrift is gewijd aan het bestuderen van symmetrische cryptografische algoritmes. Deze vormen de basis van vrijwel elk beveiligingssysteem. De doelstelling van dit proefschrift is het vergroten van de kennis over deze algoritmes door middel van het uitbreiden van het wiskundige fundament voor ontwerp, analyse en veiligheidsbewijzen van symmetrische algoritmes.

Als eerste bijdrage stellen we *niet-gladde cryptanalyse* voor, een nieuwe methode voor de cryptanalyse van symmetrische algoritmes gebaseerd op de toepassing van niet-gladde optimisatiemethodes op het oplossen van stelsels vergelijkingen over eindige velden van karakteristiek twee. We focussen verder op *rebound-aanvallen*, een krachtige recente methode voor de cryptanalyse van hashfuncties. In deze context stellen we nieuwe varianten van rebound-aanvallen op de hashfunctie Grøstl-0 voor en tonen een ontwerpstrategie aan voor hashfuncties die bestand zijn tegen rebound-aanvallen. Dit leidt tot het ontwerp van de nieuwe hashfunctie *Whirlwind*.

Voor het ontwikkelen van nieuwe ontwerpcriteria op basis van verbeterde cryptanalysemethodes is het van cruciaal belang om de kracht van deze aanvallen zo precies mogelijk te kunnen inschatten. We bestuderen twee belangrijke cryptanalysemethodes, namelijk lineaire cryptanalyse en differentiële cryptanalyse met structuren, en ontwikkelen nieuwe wiskundige modellen voor de complexiteitsanalyse van deze aanvallen die naukeuriger en realistischer zijn dan reeds gekende modellen. In beide gevallen is onze studie gebaseerd op een diepere analyse van de statistische eigenschappen die door deze aanvallen uitgebuit worden.

Deze analyse wordt aangevuld met een studie van het ideale statistische gedrag ten opzichte van lineaire en differentiële cryptanalyse. We leiden een expliciete beschrijving af van een referentiepunt voor weerstand tegen deze aanvallen.

Ten slotte richten we onze aandacht op een structurele analyse van *key-alternating cijfers*. Aangezien de alomtegenwoordige Advanced Encryption Standard (AES) tot deze categorie behoort, zijn key-alternating cijfers een bijzonder belangrijke ontwerpmethode voor blokcijfers. Wij bewijzen in dit proefschrift dat key-alternating cijfers vanuit de structurele oogpunt als een veilige ontwerpstrategie kunnen worden beschouwd. Bovendien tonen wij aan dat meerdere (maar niet té veel) ronden te hebben in zo'n ontwerp een constructief effect heeft op de weerstand tegen lineaire cryptanalyse.

Zusammenfassung

Sichere Kommunikation und Datenverarbeitung haben in der modernen Gesellschaft eine derartige Allgegenwart erreicht, dass ihre Bedeutung kaum überschätzt werden kann. Ob Privatleute, Firmen oder Regierungen — sie alle sind auf Technologien angewiesen, die die Vertraulichkeit, Integrität und Authentizität ihrer Kommunikation sicherstellen. Zur Realisierung dieser Sicherheitsziele setzt man *kryptografische Algorithmen* ein, oft sogar ohne dass die Nutzer sich dessen explizit bewusst sind.

Um sinnvoll einsetzbar zu sein, ist es wichtig, einschätzen zu können, ob und in welchem Umfang ein kryptografischer Algorithmus die beabsichtigten Sicherheitsziele erreicht. Jeglicher Fortschritt bezüglich unserer Möglichkeiten, kryptografische Algorithmen zu analysieren ("brechen") hat seinerseits ein verbessertes Verständnis der Entwurfsmöglichkeiten zur Folge, und umgekehrt. Gleichzeitig ist die Möglichkeit, mathematisch präzise Aussagen über einerseits die von Algorithmen gebotenen Sicherheitsgarantien, und andererseits die Mächtigkeit von Angriffen treffen zu können, eine wichtige Ergänzung dieses andauernden Wechselspiels zwischen Angreifern und Designern.

Die vorliegende Dissertation beschäftigt sich mit symmetrischen kryptografischen Algorithmen, welche in nahezu jedem Sicherheitssystem Verwendung finden. Sie hat ein verbessertes Verständnis dieser Algorithmen durch eine Erweiterung des mathematischen Fundaments von Entwurf, Analyse und Sicherheitsbeweisen von symmetrischen Algorithmen zum Ziel.

Als ersten Beitrag schlagen wir *nichtglatte Kryptoanalyse* vor, eine neue Methode zur Kryptoanalyse symmetrischer Algorithmen, welche auf der Anwendung nichtglatter Optimierungsverfahren zur Lösung von Boole'schen Gleichungssystemen basiert. Wir wenden uns dann den *Rebound-Angriffen* zu, einer wichtigen modernen Technik zur Kryptoanalyse von Hashfunktionen. In diesem Zusammenhang stellen wir neue Erweiterungen dieses Angriffes auf die Hashfunktion Grøstl-0 vor. Des Weiteren untersuchen wir Möglichkeiten zum Entwurf einer Rebound-resistenten Hashfunktion und schlagen darauf basierend die neue Hashfunktion *Whirlwind* vor.

Um Fortschritte in der Kryptoanalyse sinnvoll in neue Entwurfskriterien umsetzen zu können, ist es unerlässlich, die Mächtigkeit von Angriffen genau einschätzen zu können. Im Rahmen dieser Dissertation wenden wir uns hier zwei wichtigen Analysemethoden für symmetrische Algorithmen zu: der linearen Kryptoanalyse und differentiellen Angriffen mit Strukturen. Für beide Angriffe entwickeln wir ein verbessertes und realitätsnäheres Modell für die Komplexitätsanalyse, basierend auf einer vertieften Analyse der diesen Angriffen zu Grunde liegenden statistischen Eigenschaften.

Komplementär zu dieser Analyse untersuchen wir, was ideal sicheres statistisches Verhalten bezüglich linearer und differenzieller Kryptoanalyse bedeutet, und geben eine explizite Charakterisierung eines *Referenzpunktes* für das Widerstehen dieser Angriffe.

Schließlich wenden wir uns der bis dato weitgehend offenen Frage zu, was sich zur Sicherheit von sogenannten *schlüsselalternierenden Chiffren* vom strukturellen Standpunkt aus herleiten lässt. Da der universell verwendete Advanced Encryption Standard (AES) zu dieser Klasse gehört, sind schlüsselalternierende Chiffren eine besonders wichtige Konstruktionsmethode für Blockchiffren. Wir beweisen, dass schlüsselalternierende Chiffren als strukturell stichhaltiges Entwurfsprinzip gelten können. Bezüglich der Resistenz gegen lineare Angriffe etabliert unsere Analyse die Bedeutung mehrerer, aber nicht zu vieler Runden in solchen Chiffren.

Contents

Abstract iii

Contents ix

List of Figures xv

List of Tables xix

1 Introduction 1

1.1 Scope of this thesis 3

1.2 Contributions and outline 4

2 Foundations 7

2.1 Mathematical foundations 7

2.1.1 Finite fields and Boolean functions 7

2.1.2 Discrete and continuous probability distributions 8

2.2 Symmetric primitives 10

2.2.1 Stream ciphers 10

2.2.2 Block ciphers 10

2.2.3 Hash functions 12

2.3 Symmetric cryptanalysis methods 13

2.3.1 Differential cryptanalysis 13
2.3.2 Linear cryptanalysis 14

3 Stream ciphers and nonsmooth cryptanalysis 17
3.1 The equation-solving approach in cryptanalysis 18
3.1.1 Formulation as continuous optimisation problems 19
3.2 Pseudo-Boolean optimisation and the DDT heuristic 21
3.3 Nonsmooth cryptanalysis 25
3.3.1 A nonsmooth model of Boolean equations 25
3.3.2 Minimising nonsmooth functions: The Bundle method . 27
3.3.3 Dealing with nonconvex objective functions 33
3.4 Applying techniques from nonsmooth optimisation to MICKEY 36
3.4.1 The stream cipher MICKEY 36
3.4.2 A scalable variant 37
3.4.3 Equations representing Mini-MICKEY 38
3.4.4 Attack scenarios 40
3.4.5 Experimental results 41
3.4.6 The DDT heuristic and MICKEY 45
3.4.7 Discussion 45
3.5 Conclusions 49

4 Hash functions and the rebound attack 51
4.1 Wildpool: Difference propagation in unaligned diffusion layers . . 51
4.1.1 Brief description of Wildpool 52
4.1.2 Difference cancellation in unaligned diffusion layers . . . 52
4.1.3 Proving bounds 56
4.2 The rebound attack 57
4.3 Whirlwind: Designing a rebound-resistant hash function 58

4.3.1 Description of Whirlwind 59
4.4 Security analysis 62
4.4.1 On basic differential attacks 62
4.4.2 Related differentials and algebraic attacks 63
4.4.3 Rebound attacks 63
4.5 Normal basis decomposition for efficient implementations . . . 68
4.5.1 Properties of normal bases in finite fields 68
4.5.2 Normal bases in Whirlwind 69
4.5.3 Multiplication and inversion using normal bases 69
4.5.4 Basis choice for Whirlwind 70
4.5.5 Practical implementation considerations 70
4.6 Grøstl-0: Attacking a rebound-resistant hash function 75
4.6.1 Description of Grøstl-0 76
4.6.2 The internal differential attack 79
4.6.3 Variants of the rebound attack 79
4.7 Collisions for round-reduced Grøstl-0 82
4.7.1 Attack strategy 82
4.7.2 Attacks on Grøstl-0-256 83
4.7.3 Attacks on Grøstl-0-512 88
4.7.4 On memoryless collisions 91
4.8 Distinguishers for the round-reduced Grøstl-0-256 permutation 93
4.8.1 Distinguishers in the limited-birthday model 94
4.8.2 Distinguishers in the subspace problem model 95
4.9 Conclusions 95

5 Block ciphers and statistical cryptanalysis 97

5.1 Analysing the complexity of linear attacks 99

5.1.1 Problem statement 99

5.1.2 Common assumptions 102

5.1.3 Previous work 103

5.1.4 Distribution of biases in Boolean permutations 105

5.1.5 Motivation for a re-evaluation 105

5.2 An improved model for the analysis of linear attacks 106

5.2.1 More accurate wrong key randomisation 106

5.2.2 Probability of success 108

5.2.3 Non-monotonicity of success rate as function of data complexity 113

5.2.4 Evaluation of the data complexity 114

5.3 Experimental results 116

5.3.1 Experiment 1: The impact of a low bias 117

5.3.2 Experiment 2: The impact of a high advantage 118

5.3.3 Larger block sizes 121

5.4 Impact of the linear hull effect 122

5.4.1 Motivation 122

5.4.2 Decomposing linear hulls: signal and noise 124

5.4.3 An explicit algorithm for key-alternating ciphers 125

5.4.4 Experimenting the signal/noise decomposition 126

5.5 Analysing the complexity of differential attacks with structures 128

5.5.1 Motivation: Using multiple differentials in an attack . . 128

5.5.2 Previous work 130

5.6 A Model for structure attacks 132

5.6.1 Principle of the attack 132

5.6.2 Analysis of the attack 134

5.6.3 Evaluating the success probability of structure attacks . 135

5.6.4 Ratio of weak keys in a structure attack 136
5.7 Applications: Attacking PRESENT and Serpent 138
5.7.1 The block cipher PRESENT 138
5.7.2 Identifying good differential paths for PRESENT 139
5.7.3 Key recovery attack on 18-round PRESENT-80 139
5.7.4 Attack on reduced-round Serpent 141
5.8 Conclusions 143

6 Designing block ciphers: Statistical and structural aspects 145

6.1 Distributions of linear and differential properties 146
6.1.1 Distributions over all functions and permutations 147
6.2 Distributions for a randomly drawn function or permutation . . 148
6.2.1 Motivation 148
6.2.2 On the expected nature of our results 149
6.2.3 Main result 150
6.2.4 Discussion 152
6.2.5 Numerical illustration 153
6.2.6 On related results 153
6.3 High-probability approximations and differentials for a permutation 155
6.4 Impossible differentials and zero correlation approximations for a permutation 159
6.5 Structural properties of key-alternating ciphers 161
6.5.1 Previous work 161
6.5.2 The construction 162
6.6 Generic attacks 162
6.6.1 Daemen's attack for $t = 1$ 163
6.6.2 A meet in the middle attack 163

6.7 Statistical properties 164
6.7.1 Fourier coefficients and Fourier spectrum 164
6.7.2 Fourier coefficients in the single-round key-alternating cipher 165
6.7.3 Fourier coefficients in the t-round key-alternating cipher 166
6.7.4 Discussion 169
6.8 Indistinguishability analysis of key-alternating ciphers 170
6.8.1 The provable security setting 170
6.8.2 An indistinguishability bound for key-alternating ciphers 171
6.8.3 An attack meeting the conjectured bound 172
6.9 Practically instantiating the construction 174
6.9.1 AES^2: a block cipher proposal based on AES 174
6.10 Conclusions 176

7 Conclusion and outlook **177**
7.1 Summary of results 177
7.2 Open problems 179

Bibliography **181**

Curriculum vitae **197**

List of publications **199**

List of Figures

2.1 One round of a Feistel network. 11

2.2 One round of a Substitution-Permutation network. 12

3.1 Conversion of the Boolean function $F(x, y) = \overline{x \vee (x \oplus y)}$ to the reals. 27

3.2 Example illustrating the inapplicability of gradient descent search to nonsmooth objective functions. 29

3.3 The idea of the Bundle method: approximate f from below by a cutting plane model. $\tilde{x}$ denotes the minimum of f on X. 31

3.4 Determining descent directions via projection on the convex approximation of the ε-subdifferential. $G = \text{conv}\{g^0, \ldots, g^k\}$ denotes the convex hull of the subgradients in the bundle. . . . 32

3.5 Architectural overview of Mini-MICKEY. 38

3.6 Penalty terms to reward Booleanness. 44

4.1 Half-zero output difference causing deactivation of rows. 53

4.2 Modified view on Wildpool's state. The arrows indicate the operation of π. 57

4.3 $\varphi(h, m)$, the compression function of Whirlwind. 62

4.4 Basic truncated differential trail covering 4 rounds. 65

4.5 The collision attack on 4.5 rounds. 65

4.6 The near-collision attack on 5.5 rounds. 66

4.7 The near-collision attack on 6.5 rounds. 67
4.8 The compression function of Grøstl 77
4.9 IV and AddConstant in Grøstl-0-256 78
4.10 IV and AddConstant in Grøstl-0-512 78
4.11 Example path for the start-from-the-middle rebound attack. . . 80
4.12 Example path for the Super-Sbox rebound attack. 81
4.13 Truncated difference patterns for the analysis of Grøstl. 83
4.14 Differential path between P and Q of 5-round Grøstl-0-256 . . . 84
4.15 Differential path between P and Q of 6-round Grøstl-0-256 . . . 85
4.16 Differential path between P and Q of 8-round Grøstl-0-224 and -256 87
4.17 Differential path between P and Q of 6-round Grøstl-0-512 . . . 89
4.18 Differential path between P and Q of 8-round Grøstl-0-384 and -512 90
4.19 Differential path between P and Q of 9-round Grøstl-0-384 and -512 92
4.20 Differential path for the 7-round distinguisher of the Grøstl permutation. 93
4.21 Differential path for the 8-round distinguisher of the Grøstl permutation. 93

5.1 Advantage of a linear attack 100
5.2 Relation between the type I and II error probabilities, success probability P_S and advantage a. 101
5.3 Comparison of the standard and new wrong key randomisation hypotheses (WKRH). 109
5.4 Experimental distribution of the sample bias over 2^{16} wrong keys and 2^{16} texts for SmallPresent and small-scale RC6. 110
5.5 Example of an equal overlapping area between $\mathcal{N}(\epsilon, \frac{1}{4N})$ and $\mathcal{N}(0, \frac{1}{4}\left(\frac{1}{N} + \frac{1}{2^n}\right))$ for $n = 20$ and different values of N. 114

5.6 Experimental verification of non-monotonous behaviour for SmallPresent-20 with $\epsilon = 2^{-10}$ and $a = 12$. 115

5.7 Theoretical and experimental evaluation of the data complexity estimate of Proposition 1 for different levels of advantage. The cipher is SmallPresent with 8 rounds, $n = 20$, $|\epsilon| = 2^{-8.22}, P_S = 0.95$. 119

5.8 Theoretical and experimental evaluation of the data complexity estimate of Proposition 1 for RC6-5/8/10 and different levels of advantage. $n = 20$, $|\epsilon| = 2^{-8.09}, P_S = 0.95$. 119

5.9 Theoretical and experimental evaluation of the data complexity estimate of Proposition 1 for higher levels of advantage. The cipher is SmallPresent with 8 rounds, $n = 20, \epsilon = 2^{-8}, P_S = 0.95$. 120

5.10 Theoretical and experimental evaluation of the data complexity estimate of Proposition 1 for RC6-5/6/10 and higher levels of advantage. $n = 20, \epsilon = 2^{-7.88}, P_S = 0.95$. 120

5.11 Evaluation of the data complexity estimate of Proposition 1 for bigger block length and marginal bias. $n = 64, \epsilon = 1.49 \cdot 2^{-32}, P_S = 0.5$. 121

5.12 Evaluation of the data complexity estimate of Proposition 1 for bigger block length and higher levels of advantage. $n = 64, \epsilon = 1.49 \cdot 2^{-30}, P_S = 0.5$ 123

5.13 Theoretical and experimental evaluation of the data complexity with the signal-noise decomposition of Alg 2. Cipher is SmallPresent with 5 rounds, $n = 20, P_S = 0.95$, the signal part contains 8 trails U_j with $2^{-10} \leq |\epsilon_{U_j}| \leq 2^{-11}$. Experimental value of ϵ is $2^{-8.02}$. 128

5.14 Pairs of plaintext organised in a quartet. 129

5.15 One round of the PRESENT block cipher. 138

5.16 The block cipher Serpent reduced to 8 rounds. 141

6.1 Experimental distribution of the imbalance for a single randomly drawn permutation and its normal approximation. 153

6.2 Experimental distribution of the cardinality of differentials for a single randomly drawn permutation and its Poisson approximation. 154

6.3 One-round key-alternating cipher: The Even-Mansour construction 162

6.4 Key-alternating cipher with t rounds 162

6.5 The indistinguishability model for the key-alternating cipher [33]. 171

List of Tables

3.1 State recovery for n bit registers and k bit key. Dimension is $2n$. 46

3.2 Key recovery for n bit registers and k bit key. Dimension is k. . 47

3.3 Guess k_0 out of 80 key bits and determine the remaining key bits for the full MICKEY ($n = 100$ bit registers and $k = 80$ bit key). Dimension is $80 - k_0$. Balancing was used in all experiments. . 48

3.4 Experimental results applying the DDT heuristic to the three attack scenarios. 49

4.1 Overview of the normal bases decomposition in Whirlwind. . . . 71

4.2 Performance figures for software implementations of Whirlwind on Intel processors. All numbers are given in cycles per byte (cpb). 73

4.3 Summary of results for Grøstl-0. "coll.", "FSC", "SFSC", and "dist." are abbreviations for collision, free-start collision, semi-free-start collision, and distinguisher, respectively. 76

4.4 Input pair (a, b) of Grøstl's P permutation following the trail used in the 7-round distinguisher. 94

5.1 Theoretical and experimental weak key ratio for SmallPresent-24. 138

5.2 Comparison of our attacks on PRESENT with the multiple differential cryptanalysis of [27]. 141

5.3 Comparison of our structure attacks to previous differential attacks on Serpent. 143

6.1 Numerical illustration of propositions 6.9 and 6.11. (α, β) denotes either a linear approximation or a differential for the randomly drawn n-bit permutation π_0. 158

6.2 Probability of a zero correlation approximation in a randomly drawn n-bit permutation π_0 for common values of n. 160

6.3 Practical performance of AES^2 using AES-NI instructions. . . . 175

Chapter 1

Introduction

> Sicher ist, dass nichts sicher ist.
> Selbst das nicht.
>
> Joachim Ringelnatz, 1883–1934

Communication, the interchange of information, has always been a central aspect of human society. While direct oral or non-verbal communication from person to person is subject only to social barriers, the historical development of indirect forms of communication — from writing systems to telecommunication, culminating in the development of computer networks — has soon led to certain unavoidable complications. Already in the simple example of sending a hand-written letter, the intended communication with the recipient can go wrong in a number of ways that are of no concern in the case of direct personal communication: Will anybody except the intended recipient read the letter? Will the letter actually be delivered to the correct recipient? Could its contents be changed in transit, either accidentally or with malicious intent? In modern terms, indirect communication makes it harder to ensure *confidentialty*, *authenticity* or *integrity* of messages.

These concerns very naturally lead to the development of techniques alleviating them. Traditionally, problems with authenticity and integrity were usually solved by trusted messengers, while the confidentiality issue was adressed by algorithmic and/or technological means. This is where the name *cryptography* stems from: the practice of secret writing. The contemporary use of the term *cryptography* also includes techniques for authenticity and integrity of communications. Modern cryptography captures the objective of achieving confidentiality in the notion of *cryptosystems*:

Definition 1.1 (Cryptosystem)**.** A *cryptosystem* is a tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ satisfying the following properties:

1. $\mathcal{P}$ is the (finite) set of *plaintexts*;
2. $\mathcal{C}$ is the (finite) set of *ciphertexts*;
3. $\mathcal{K}$ is the (finite) set of *keys*;
4. $\mathcal{E} = \{E_k : k \in \mathcal{K}\}$ is a family of *encryption functions* $E_k : \mathcal{P} \to \mathcal{C}$;
5. $\mathcal{D} = \{D_k : k \in \mathcal{K}\}$ is a family of *decryption functions* $D_k : \mathcal{C} \to \mathcal{P}$;
6. For all $e \in \mathcal{K}$ there exists a $d \in \mathcal{K}$ such that we have

$$D_d(E_e(p)) = p \quad \text{for all plaintexts } p \in \mathcal{P}. \tag{1.1}$$

The key idea here is that only the knowledge of the decryption key corresponding to the key used for the encryption of a plaintext into its ciphertext determines which element of the function family $\mathcal{D}$ has to be used to obtain the plaintext again. If we always have $d = e$ in (1.1), one speaks of a *symmetric-key* or simply *symmetric* cryptosystem. In this case, two parties wishing to communicate using the cryptosystem have to agree a-priori which shared secret key they are going to use. We will describe symmetric algorithms in more detail in Section 2.2.

An important second class of cryptosystems is defined by the case that d cannot be derived from e with feasible computational effort. In this case, the encryption key e can be public, while the decryption key has to be kept secret. These are called *public-key* or *asymmetric-key* cryptosystems.

The study of how to attack a cryptographic algorithm is called *cryptanalysis*. In this context, the security of a cryptosystem can be defined in different ways [124, 144]. In the information-theoretic sense, “secure” means *unconditionally secure*: Even with unlimited computational resources, no adversary can break the scheme.

If a scheme is not unconditionally secure, it is always possible to break it by what is known as *brute force*: simple exhaustive search.

In the complexity-theoretic sense, “secure” means security against an adversary performing arbitrary computations within certain limits (ideally, exponential in the size of the key space). Lastly, in the practical sense of the word, a scheme can be considered secure if there is no known way of breaking it faster than exhaustive search. While this interpretation is necessarily somewhat loose, it has led to the development of practically usable algorithms that are still considered secure after years of cryptanalysis [57].

In order to make a clear distinction between *cryptography* and *cryptanalysis*, the term *cryptology* is sometimes used for the "combined" science of devising and analysing cryptographic algorithms. We will however follow the common convention and use *cryptography* and *cryptology* interchangeably [124].

1.1 Scope of this thesis

In the past decades, significant progress has been made in providing solid foundations for the analysis and design of symmetric cryptographic algorithms. The goal of obtaining rigorous arguments for a certain scheme has been pursued from both the practical and the complexity-theoretic point of view. In the practical security approach, methods have been developed to prove that concrete algorithms resist certain classes of attacks [56, 117, 135, 158]. This implies that in order to obtain even only a slight advantage over exhaustive search for contemporary symmetric algorithms such as the AES [57], the invention of new a cryptanalytic technique is required [31]. On the other hand, the complexity-theoretic approach has led to generic designs for a secure cipher [113], and also concrete proposals provably basing their security on a hard computational problem [49].

However, the current level of understanding still leaves a lot to be desired, especially concerning the analysis of generalised constructions, as opposed to concrete algorithm proposals.

In this context, this thesis aims at studying the interplay between new cryptanalysis techniques and the design of symmetric algorithms. Additionally, formal security arguments against known attacks are analysed based on more rigorous mathematical models, and special focus is given to the question what the insights gained from proving security for concrete algorithms can tell us about the security of generalised constructions, i.e. whole families of algorithms.

1.2 Contributions and outline

We review some basic definitions and techniques in Chapter 2. The first part of the thesis then explores the impact of new attack techniques on design strategies (and vice versa) at the example of algorithms for solving non-linear equations over finite fields of characteristic two. In theory, any cryptographic algorithm can be broken by solving a corresponding system of nonlinear equations. In Chapter 3, we present a novel approach to solving these equations over the real domain by modeling them as a continuous optimisation problem. Applying it to the stream cipher MICKEY, we analyse advantages and limitations of this approach and compare it to classical pseudo-Boolean programming. This work has been published in [156].

In contrast to this continuous equation-solving strategy, Chapter 4 deals with a dedicated discrete technique for solving typical non-linear equations arising in cryptographic applications, the Rebound technique [122]. We outline the power of this technique when applied to a cryptographic hash function secure against basic differential collision attacks and analyse which design changes are necessary to counter this cryptanalysis technique, leading to the hash function Whirlwind. Finally, we demonstrate that a new variant of the Rebound technique can improve its applicability even to the hash function Grøstl-0 which was designed to resist this attack. This cryptanalysis in turn prompted design changes, which lucidly demonstrates the importance of the interplay between advances in cryptanalysis and design. The Whirlwind hash function proposal is a joint work with Paulo Barreto, Ventzislav Nikov, Svetla Nikova and Vincent Rijmen, and has been published in [11]. The results on Grøstl-0 were developed in collaboration with Kota Ideguchi and Bart Preneel and have been published in [90] and [91].

Some of the most powerful cryptanalytic techniques in symmetric cryptography are statistical in nature, and many proposed attacks have time and data complexities lower than exhaustive search, but beyond what can be practically implemented and verified. Being able to accurately estimate their complexities is therefore of great importance to the evaluation of symmetric algorithms. In Chapter 5, we propose extended mathematical models for the complexity analysis of both linear cryptanalysis and differential cryptanalysis with structures by taking a more complete statistical picture into account. The work on linear cryptanalysis was done jointly with Andrey Bogdanov and is currently in submission [36]. The results on differential cryptanalysis are a joint work with Meiqin Wang, Yue Sun and Bart Preneel and published in [161].

In Chapter 6, we take a generic look at the design of block ciphers. For any key, a block cipher should behave like a random permutation. By obtaining a statement about the distribution of linear and differential properties in a randomly chosen permutation, we are able to define a *reference point* for statistical attacks. We then turn to the question of constructing a block cipher from a small set of public permutations. Specifically, we provide a theoretical analysis of key-alternating block ciphers with regard to their statistical properties, quantifying the constructive effect of having multiple, but not too many, rounds in such a design. Finally, the generic resistance against information-theoretic adversaries limited to a certain number of queries to the permutations and the cipher is analysed. This is a joint work with Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, François-Xavier Standaert and John Steinberger and has been published in [33, 34].

Chapter 7 concludes and lists some starting points for future work.

Chapter 2

Foundations

2.1 Mathematical foundations

2.1.1 Finite fields and Boolean functions

Finite fields are a fundamental mathematical structure for symmetric-key cryptography. Nearly all algorithms and operations are defined on $\mathbb{F}_2$, the finite field of order 2, or some field extension thereof.

For every prime p and every positive integer n, there exists a finite field with p^n elements. We denote the finite field of order p^n by $\mathbb{F}_{p^n}$ or equivalently as $\mathrm{GF}(p^n)$.

Let $q = p^n$ be any prime power (including the case $n = 1$). The finite field $\mathrm{GF}(q^m)$ is isomorphic to an m-dimensional vector space over its subfield $\mathrm{GF}(q)$. This means that one can construct a *basis* for $\mathrm{GF}(q^m)$ over $\mathrm{GF}(q)$. Any such basis consists of m elements $\beta_0, \beta_1, \ldots, \beta_{m-1} \in \mathrm{GF}(q^m)$ such that all elements of $\mathrm{GF}(q^m)$ can be written as a linear combination of the elements β_j, with all coefficients being elements of $\mathrm{GF}(q)$. There are many different choices possible for the basis.

Viewing the field extension $\mathrm{GF}(q^m)$ as a vector space over $\mathrm{GF}(q)$, the *trace* of an element $a \in \mathrm{GF}(q^m)$ over $\mathrm{GF}(q)$ is defined as

$$\mathrm{Tr}_{\mathrm{GF}(q^m)/\mathrm{GF}(q)}(a) \overset{\mathrm{def}}{=} a + a^q + \cdots + a^{q^{m-1}}.$$

If q is prime, this trace is called the absolute trace of a.

We will often deal with $\mathbb{F}_2 = \{0, 1\}$, the finite field with two elements and the n-dimensional vector space over $\mathbb{F}_2$, denoted $\mathbb{F}_2^n$. The canonical scalar product of two vectors $a, b \in \mathbb{F}_2^n$ is denoted by $a^T b$.

A function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ with domain $\mathbb{F}_2^n$ and range $\mathbb{F}_2$ is called a *Boolean function.* A mapping $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is called a *vectorial Boolean function.* We usually omit the prefix “vectorial” if it is clear from the context. In the case of a vectorial Boolean function with $n = m$, we speak of a *Boolean transformation.* A special case of transformations are *Boolean permutations*, bijective mappings of $\mathbb{F}_2^n$. We denote the set of all such permutations by S_n, the symmetric group of degree n.

A fundamental tool in the theory of Boolean functions is the *Walsh-Hadamard* transform, the discrete Fourier transform for fields of characteristic two. It is defined for functions with domain $\mathbb{F}_2^n$ and range $\mathbb{R}$ (note that this includes a Boolean domain by field embedding). We denote the set of all such functions by $\mathbb{R}^{\mathbb{F}_2^n}$. The *Walsh-Hadamard transform* (*Walsh transform* for short)

$$\mathcal{W} : \mathbb{R}^{\mathbb{F}_2^n} \to \mathbb{R}^{\mathbb{F}_2^n}, \qquad f \mapsto \widehat{f}$$

is then defined as

$$\mathcal{W}(f)(u) = \widehat{f}(u) := \sum_{x \in \mathbb{F}_2^n} f(x) \cdot (-1)^{u^T x}. \tag{2.1}$$

The Walsh transform $\widehat{\vartheta_f}$ of f’s *characteristic function* $\vartheta_f : \mathbb{F}_2^n \times \mathbb{F}_2^q \to \mathbb{R}$, $\vartheta_f(x, y) = 1 \Leftrightarrow y = f(x)$, is called the *Walsh* or *Fourier spectrum* of f, its value at point (α, β) the *Walsh* or *Fourier coefficient* of f at (α, β). We denote this value by

$$W^f_{\alpha,\beta} \overset{\mathrm{def}}{=} \sum_{x \in \mathbb{F}_2^n} (-1)^{\alpha^T x + \beta^T f(x)}.$$

2.1.2 Discrete and continuous probability distributions

We denote by $\mathrm{Bern}(p)$ the Bernoulli distribution with success probability p, by $\mathrm{Bin}(N, p)$ the binomial distribution with N experiments and success probability p, and by $\mathcal{N}(\mu, \sigma^2)$ the normal distribution with mean μ and variance σ^2. The probability density and cumulative distribution function of the standard normal distribution $\mathcal{N}(0, 1)$ are denoted by $\phi(x)$ and $\Phi(x)$, respectively. For a normal random variable X, the distribution of $|X|$ follows the folded normal distribution [110] with probability density function $f_\phi(x)$ and cumulative distribution function $F_\Phi(x)$.

By $X \sim_v \mathcal{D}$, we denote a random variable X following a distribution $\mathcal{D}$ taken over all values of v. The expectation of X with respect to v is denoted by $\mathbf{E}_v[X]$, its variance (with respect to v) by $\mathbf{Var}_v[X]$.

If the probability distribution of a discrete random variable X taking on integer multiples of ϵ is approximated by a continuous distribution with cumulative distribution function $D(x)$, a *continuity correction* is required, which yields [60]

$$\Pr(X = x) \approx \epsilon \frac{dD(x)}{dx},$$

$$\Pr(X < x) \approx D(x - \epsilon/2).$$

The *Central limit theorem (CLT)* states under which conditions the mean of a sufficiently large number of independent random variables will be approximately normally distributed. There are many variants of the central limit theorem. We will use the Chebyshev formulation of the CLT:

Theorem 2.1 (Chebyshev's CLT, [154, p. 488]). *Let $\{X_n\}_{n=1}^{\infty}$ be a sequence of independent random variables satisfying the following conditions:*

1. $\mathbf{E}[X_k]^2 < \infty$ *for* $k = 1, 2, \ldots$ *and*

$$\lim_{n\to\infty} \Pr(|X_n| < b) = 1 \textit{ for } b > 0,$$

$$s_n^2 := \mathbf{Var}[\sum_{i=1}^{n} X_i] \longrightarrow \infty \textit{ as } n \to \infty;$$

2. $\mathbf{E}[X_k] = \mu_k$, $\mathbf{Var}[X_k] = \sigma_k^2$, $k = 1, 2, \ldots$.

Then it holds that

$$\lim_{n\to\infty} \Pr\left(\left[\frac{1}{s_n}\sum_{k=1}^{n}(X_k - \mu_k)\right] \le z\right) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{z} e^{-\frac{1}{2}u^2} du$$

for all $z \in \mathbb{R}$.

2.2 Symmetric primitives

Symmetric algorithms include symmetric-key encryption algorithms (cryptosystems as defined in Chapter 1), and cryptographic hash functions. Symmetric encryption algorithms can usually be categorized as either *stream* or *block ciphers*; hash functions can be keyed (message authentication codes) or unkeyed. This thesis deals with stream ciphers, block ciphers, and unkeyed hash functions. They will be described in turn.

2.2.1 Stream ciphers

Stream ciphers are symmetric encryption algorithms that encrypt by combining the plaintext with a pseudo-random sequence. They typically operate on small units of plaintext at a time, usually bits or bytes. Stream ciphers take a key k and an initial value IV to generate a pseudo-random sequence called the *key stream* which is used to encrypt and decrypt data using the bitwise XOR operation. The IV is used as a randomisation parameter to be able to generate multiple key streams from the same key. Contrary to the key, it does not have to be kept secret.

Stream ciphers come in two flavours: *synchronous* or *self-synchronising*. Self-synchronising stream ciphers generate the key stream based on k, the IV and some feedback from the last couple of ciphertext bits. By contrast, the key stream generated by synchronous stream ciphers is completely defined by k and the IV only and independent of the plaintext. Synchronous stream ciphers are far more common than self-synchronising, some important examples being the ubiquitous RC4 or the eSTREAM finalists, e.g. Trivium [42, 43].

2.2.2 Block ciphers

A *block cipher* is a mapping $E : \mathbb{F}_2^n \times \mathbb{F}_2^\kappa \to \mathbb{F}_2^n$ with the property that $E_k \overset{\text{def}}{=} E(\cdot, k)$ is a bijection of $\mathbb{F}_2^n$ for every $k \in \mathbb{F}_2^\kappa$. If $y = E_k(x)$, we refer to x as the *plaintext*, k as the *key* and y as the *ciphertext* of x under the key k. We call n the *block length* and κ the *key size* of the cipher. An alternative interpretation is to view a block cipher as a family of permutations of $\mathbb{F}_2^n$ indexed by the key.

Block ciphers are often constructed as iterated mappings based on *round functions* $\rho_i[k_i]$. Let R denote the number of rounds. A *key scheduling algorithm* expands the encryption key k into R round keys $K \overset{\text{def}}{=} (k_0, \ldots, k_{R-1})$. The ciphertext y is then obtained as $y = x_R$ with $x_{i+1} = \rho_i[k_i](x_i)$. If the

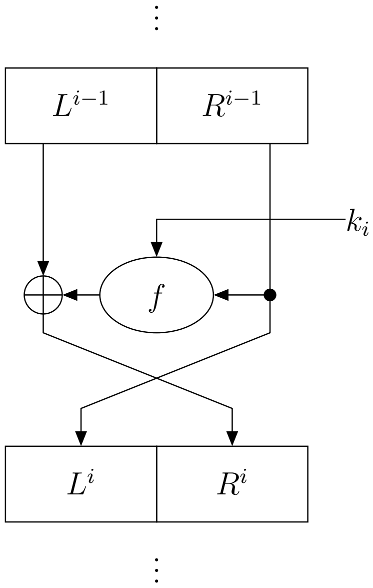


Figure 2.1: One round of a Feistel network.

iteration can be written as a sequence of unkeyed rounds and bitwise addition of the round keys by XOR, the cipher is called a *key-alternating cipher* [54, 57]. If all round functions ρ_i are identical, we speak of an *iterated* block cipher.

Two widespread paradigms for the construction of iterated block ciphers are *Feistel networks* and *Substitution-permutation networks (SPN)s*, depicted in Figures 2.1 and 2.2 , respectively. Note that ciphers using the SPN construction are key-alternating by definition. However, also some Feistel ciphers [71] can be written as key-alternating ciphers [58].

In an SPN, one often uses a linear mapping f_p to provide diffusion. This has become increasingly common since the invention of the wide trail design strategy [55–57, 144]. Let $n = ml$. The diffusion performance of such a linear mapping $f_p : \mathbb{F}_2^{ml} \to \mathbb{F}_2^{ml}$ can be expressed by means of its *branch number*

$$\mathcal{B}(f_p) := \min_{0 \neq a \in \mathbb{F}_2^{ml}} w(a) + w(f_p(a)),$$

with w denoting the *weight* (number of nonzero components) of the l-component vector a. We note that $\mathcal{B}(f_p) \leq l + 1$.

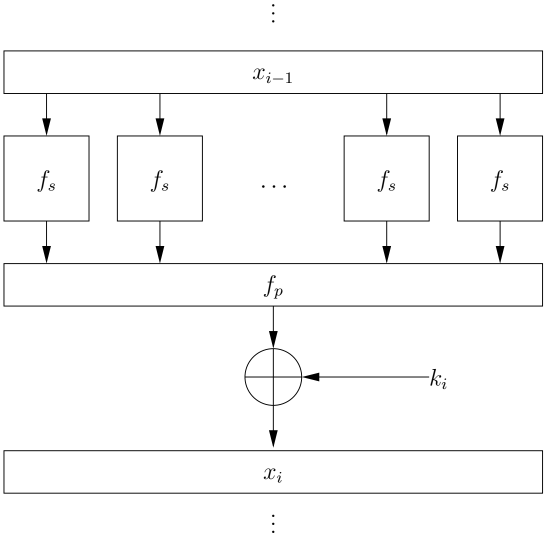


Figure 2.2: One round of a Substitution-Permutation network.

2.2.3 Hash functions

In this thesis, we deal with *unkeyed* hash functions only. An unkeyed cryptographic hash function is a function $h : \mathbb{F}_2^* \to \mathbb{F}_2^n$ mapping sequences of arbitrary length to a fixed-length output. A cryptographically secure hash function should fulfill the following three informal security properties:

- Collision resistance: Finding any two inputs $m \neq m'$ such that $h(m) = h(m')$ should require at least $2^{n/2}$ operations;
- Second preimage resistance: Given m such that $h(m) = v$, finding a second input $m' \neq m$ such that $h(m) = h(m')$ should require at least 2^n operations;
- Preimage resistance: Given any hash value v, finding an input m such that $h(m) = v$ should require at least 2^n operations.

In the scope of this thesis, we will deal with collision resistance only. Like block ciphers, hash functions are often constructed in an iterated way by utilising a *compression function* ϕ taking fixed-length inputs. For a message of t fixed-length blocks, one sets $h_0 := IV$ and $h_i = \phi(h_{i-1}, m_i)$ for $i = 1, \dots, t$, where m_i denotes the i-th block of the input and IV is some fixed initial value. The hash value is then h_t or some function thereof.

For iterated hash functions, one can consider two weaker attack scenarios: A *free-start collision* is a pair of (m, I) and (m', I') with $m \neq m'$ and $I \neq I'$ such that m and m' collide when I resp. I' are used as initial values. In a *semi-free-start collision*, $I = I'$ (but can be different from the specified IV).

2.3 Symmetric cryptanalysis methods

Cryptanalytic attacks can greatly vary in their requirements regarding the attacker's abilities in the interaction with the communicating parties. Attacks on symmetric cryptosystems are typically classified as follows [124]:

Ciphertext-only. The attacker only knows a set of ciphertexts.

Known plaintext. The attacker has plaintexts $P_1, \dots, P_s$ and the corresponding ciphertexts $C_1, \dots, C_s$ at their disposal.

Chosen plaintext. Prior to the attack, the attacker *chooses* fixed plaintexts $P_1, \dots, P_s$ and *obtains* their corresponding encryptions $C_1, \dots, C_s$.

Adaptively chosen plaintext. In an extension of the chosen-plaintext setting, the attacker can choose the plaintexts *interactively* during the attack, that is, he chooses P_1, receives C_1, chooses P_2, receives C_2, and so on.

In all cases, the attacker's goal is to either derive the decryption key or to be able to perform operations that should only be possible with knowledge of the key, for instance determining the encryptions or decryptions of messages previously unknown to the attacker.

In this section, we briefly describe the theoretical foundations of the two most important statistical analysis methods for symmetric primitives. We will turn back to these attacks in more detail in Chapters 4 to 6.

2.3.1 Differential cryptanalysis

Differential cryptanalysis was proposed by Biham and Shamir [22]. To date, it is one of the most influential analysis methods for symmetric primitives, for ciphers as well as for hash functions.

Its main idea is to analyse the propagation of *differences* through mappings. Throughout this thesis, we will deal with XOR differences between Boolean vectors: if $u \oplus v = \alpha$, then the difference between u and v is said to be α.

The propagation of differences through mappings is captured in the notion of *differentials*: A differential over an n-bit vectorial Boolean function consists of an *input difference* α and an *output difference* β and is denoted by (α, β). Furthermore, we denote by $N^f_{\alpha,\beta}$ the *cardinality* of the differential (α, β) for f, which is defined as the number of unordered pairs $\{v, u\}$ with input difference α and output difference β: $N^f_{\alpha,\beta} \overset{\text{def}}{=} \{\{v, u\} \mid v \oplus u = \alpha \text{ and } f(v) \oplus f(u) = \beta\}$. The probability $\mathrm{DP}^f_{\alpha,\beta}$ of the differential (α, β) over f is related to its cardinality via $\mathrm{DP}^f_{\alpha,\beta} = N^f_{\alpha,\beta}/2^{n-1}$.

For iterated mappings $f = f^t \circ \cdots \circ f^1$, a differential (α, β) gives rise to at least t connected differentials for each step function. A *differential trail*, *characteristic* or *path* through f is a vector $Q = (\alpha_0, \dots, \alpha_t)$ with α_i specifying the difference at the i-th intermediate step. The differential probability of Q now is the fraction of unordered pairs satisfying each intermediate difference. By contrast, the differential (α_0, α_t) over the iterated map f encompasses all unordered pairs with input difference α_0 and output difference α_t, with arbitrary intermediate differences. Therefore, the probability of a differential will always be at least as high as that of a trail belonging to the differential.

In an iterated symmetric algorithm based on substitution boxes, we call an S-boxes (differentially) *active* if it has an nonzero input difference.

2.3.2 Linear cryptanalysis

Linear cryptanalysis was proposed by Matsui [115, 116]. Alongside differential cryptanalysis, it is one of the most powerful methods for the analysis of symmetric algorithms.

Its main idea is to exploit the presence of *linear approximations* (linear relations of selected input and output bits) that occur with some *bias*, i.e. their probability is different from 1/2.

A *linear approximation* (α, β) of a vectorial Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is said to hold with probability $p \overset{\text{def}}{=} \Pr_{x \in \mathbb{F}_2^n}(\alpha^T x = \beta^T f(x))$. The deviation of p from $1/2$ is called the *bias* $\epsilon \overset{\text{def}}{=} p - 1/2$. The *correlation* of a linear approximation (α, β) is $C_{\alpha,\beta} \overset{\text{def}}{=} 2p - 1 = 2\epsilon$. The quantity $LP \overset{\text{def}}{=} C^2_{\alpha,\beta}$ is called the *linear probability* of (α, β). Another related quantity is the *imbalance* of the linear approximation, which is given by $I_{\alpha,\beta} = 1/2\left(\#\{x \mid \alpha^T x \oplus \beta^T f(x) = 0\} - \#\{x \mid \alpha^T x \oplus \beta^T f(x) = 1\}\right) = 2^{n-1} \cdot$

$C_{\alpha,\beta}$. This also establishes the following relation to the Walsh transform:

$$2^n \cdot C_{\alpha,\beta} = 2 \cdot I_{\alpha,\beta} = \sum_{x \in \mathbb{F}_2^n} (-1)^{\alpha^T x + \beta^T f(x)} = W_{\alpha,\beta}.$$

As in differential cryptanalysis, a linear approximation (α, β) for an iterated map can actually be decomposed into connecting linear approximations for the intermediate steps. For each fixed value of the intermediate masks α_i, such a sequence is called a *linear trail* [57] or *linear characteristic* [115, 116]. The approximation (α, β) can permit many trails with the same input mask α and output mask β, but different intermediate masks. The collection of all such trails is called the *linear hull* (α, β) [133, 134].

Chapter 3

Stream ciphers and nonsmooth cryptanalysis

In contrast to public-key cryptography, where the security of a scheme is usually based on an—ideally tight—reduction or even equivalence to a well-known "hard" computational problem, the security of symmetric encryption algorithms is based on withstanding continuous cryptanalytic evaluation. Therefore, the evolution of cryptanalytic techniques has great impact on commonly accepted construction criteria for symmetric primitives. A prominent example of this evolution are differential and linear cryptanalysis and their refinements, which were successfully applied to many symmetric primitives, while as well motivating research into construction strategies providing resistance against these attacks.

In this chapter, we present a novel approach to the cryptanalysis of symmetric algorithms based on nonsmooth optimisation. We develop this technique as a novel way of dealing with nonlinearity over $\mathbb{F}_2$ by modeling the equations corresponding to the algorithm as a continuous optimisation problem that avoids terms of higher degree. The resulting problems are not continuously differentiable, but can be approached with techniques from nonsmooth analysis, the crucial property being that the resulting expressions are still Lipschitz-continuous. To the best of our knowledge, this is the first application of methods from nonsmooth optimisation in cryptanalysis.

While the approach presented in this chapter is applicable to generally all kinds of symmetric primitives (block ciphers, hash functions, message authentication codes), we focus on its application to the cryptanalysis of stream ciphers, more specifically the stream cipher MICKEY, which is part of the eSTREAM final portfolio.

Stream ciphers are a particularly attractive target for the numerical cryptanalysis approach since typically one bit of keystream is generated per state update, whereas block ciphers and hash functions commonly apply many iterations of a transformation to each plaintext or message block, which results in larger systems of equations. Analyzing the stream cipher MICKEY is particularly interesting since it has a relatively small state size of 200 bits and is the only algorithm in the hardware-oriented eSTREAM portfolio without any published cryptanalytic results.

First, we review the approach of cryptanalysing symmetric algorithms by equation solving. In general, direct attempts to solving the resulting equations have had very limited success; see for instance [46] for a comprehensive treatment in the case of contemporary block ciphers. We then turn to the idea of reformulating the arising discrete optimisation problems as continuous ones. In symmetric cryptanalysis, this approach has met with considerably more success than the discrete methods [37–39, 127]. However, when applied to full-scale stream ciphers such as Trivium [42, 43], their complexity greatly exceeded the effort of brute-force key search.

Since all of these methods were using polynomial models for Boolean equations, and issues with the inherently high degree of the expressions were encountered, it seems promising to investigate alternative approaches.

One of the classical frameworks in the context of solving discrete problems in a continuous way is *pseudo-Boolean optimisation* [40]. However, we prove that its principal algorithm, the *DDT* heuristic, generally does not approximate the optimum by a constant factor, making it unsuitable for our purposes.

We then develop our main tool, which we call *nonsmooth cryptanalysis*. Experimental results incidate that our method can solve instances corresponding to the full MICKEY stream cipher, although with time complexity slightly greater than brute force. These results have been published in [156].

3.1 The equation-solving approach in cryptanalysis

As seen in Chapter 2, the operation of symmetric encryption algorithms (both block and stream ciphers) can be viewed as the action of Boolean functions on vector spaces over the binary field $\mathbb{F}_2$. If an attacker is able to efficiently solve these equations (either deterministically or with non-negligible success probability), he can break the corresponding cryptographic algorithm: In the case of block ciphers, stream ciphers or message authentication codes, this enables a known-plaintext attack by solving for the variables representing the key; for hash functions, obtaining a characterisation of the set of solutions will enable the attacker to construct collisions or even (second) preimages. It is worth noting that even the ability to solve a particular subsystem or simplified variant of the system of equations – for example with some variables assumed to have particularly useful constant values – might lead to significant cryptanalytic progress, since then all plaintexts, keys or to-be-hashed messages exhibiting this very structure will be susceptible to attack.

With nonlinearity being a fundamental requirement for a cryptographic primitive, the resulting systems of equations are typically highly nonlinear and hence difficult to solve directly. Solving even quadratic equations over finite fields is known to be an NP-complete problem, and approaches based on Gröbner bases [46] or linearisation techniques [50] have had limited success so far, especially when applied to full algorithms, as opposed to reduced and/or simplified variants.

On the other hand, there exists a rich theory and supporting experimental evidence for solving real-valued systems of equations or optimising continuous objective functions [76, 131]. It therefore seems promising to apply the well-established algorithms from numerical optimisation to the cryptanalysis of symmetric primitives. This approach has been introduced by [127], considering polynomial models of the Boolean equations. Alternative approaches such as Mixed Integer Programming or Simulated Annealing have been investigated in [39] and [38].

3.1.1 Formulation as continuous optimisation problems

Since numerical optimisation algorithms generally operate on (vector spaces over) the reals, a natural approach to applying them to systems of Boolean equations is to represent these systems as equations over the reals. First of all, the two Boolean values have to be mapped to two distinct real numbers, two natural mappings being the following [37, 127].

Definition 3.1 (Standard representation). The mapping from $\{\texttt{False}, \texttt{True}\} \cong \mathbb{F}_2$ to $\mathbb{R}$ given by

$$\texttt{False} \mapsto 0,$$

$$\texttt{True} \mapsto 1$$

is called the *standard representation* of the Booleans.

Definition 3.2 (Fourier representation)**.** The mapping from $\{\texttt{False}, \texttt{True}\} \cong \mathbb{F}_2$ to $\mathbb{R}$ given by

$$\texttt{False} \mapsto 1,$$

$$\texttt{True} \mapsto -1$$

is called the *Fourier representation* of the Booleans.

By exchanging `False` and `True` in these mappings, we obtain the *dual* and *sign* representations, respectively [37,127].

There also exist efficient conversion mappings between these representations. Having fixed the conversion method, any Boolean function can be expressed in terms of polynomials with real coefficients by means of polynomial interpolation of its truth table [37]. Choosing different representations for elements and Boolean operations can result in significantly different systems of equations exhibiting different numerical behaviour [127].

A clear advantage of polynomial representations of Boolean functions is the immediate applicability of well-established equation solving techniques. However, a drawback of the polynomial approach is that addition in $\mathbb{F}_2$ is necessarily converted into a nonlinear operation: for instance, using Fourier representation, $a+b$ over $\mathbb{F}_2$ becomes $a \times b$ over the reals. Higher degrees usually have negative impact on the effectiveness of optimisation algorithms [37, p. 133].

While approaches such as the adapted standard conversion method exist to mitigate this effect to some extent [127], the approach proposed here seeks to avoid higher degrees altogether by means of a nonsmooth model of Boolean equations.

Note that instead of solving a system of equations $F(x) = 0$ with $F : \mathbb{R}^m \to \mathbb{R}^n$, one can alternatively solve the optimisation problem

$$\min_{x \in \mathbb{R}^m} \|F(x)\|, \tag{3.1}$$

where $\|\cdot\|$ is any norm, since for all y, $\|y\| \geq 0$ and $\|y\| = 0$ if and only if $y = 0$. Therefore, both numerical algorithms for solving equations and minimisation algorithms from nonlinear optimisation can be used. Since we are interested in solutions that can be mapped back to Boolean values, additional *box constraints* of the form $x_i \in [0, 1]$ can be added.

Furthermore, unless the cipher has equivalent keys (which would be a weakness on its own), the systems of equations are expected to have precisely one solution over $\mathbb{F}_2$ which is not the case for optimisation problems in general.

3.2 Pseudo-Boolean optimisation and the DDT heuristic

The idea of modeling discrete optimisation problems as continuous ones and subjecting the real-valued models to techniques inspired from nonlinear optimisation has previously been applied in the context of so-called *pseudo-Boolean functions* [81]. While this area is mostly aiming at efficient dedicated heuristics for NP-hard problems, the possibility of applying methods of convex analysis and nonlinear programming in general to problems which are otherwise discrete in nature is explicitly mentioned in [40].

We briefly outline the main concepts of pseudo-Boolean optimisation. Let $\mathbb{B} = \{0, 1\}$ and denote the unit interval by $\mathbb{U} = [0, 1]$. Functions of the type $f : \mathbb{B}^n \to \mathbb{R}$ are called *pseudo-Boolean functions*. They can be uniquely written as multi-linear polynomials, that is, as

$$f(x_1, \ldots, x_n) = \sum_{S \subseteq \{1,\ldots,n\}} c_S \prod_{j \in S} x_j \tag{3.2}$$

with $c_S \in \mathbb{R}$. If one allows both the variables x_i and their complements $\overline{x_i} \overset{\text{def}}{=} 1 - x_i$ to appear as literals, pseudo-Boolean functions can alternatively be written as *posiforms*, i.e. as

$$\phi(x_1, \ldots, x_n) = \sum_{T \subseteq \mathbf{L}} a_T \prod_{u \in T} u, \tag{3.3}$$

with $\mathbf{L} = \{x_1, \overline{x_1}, \ldots, x_n, \overline{x_n}\}$ and nonnegative coefficients, i.e. $a_T \geq 0$ if $T \neq \emptyset$. For $T = \emptyset$, a_T can be negative, and by convention, $\prod_{u \in \emptyset} u = 1$.

The following property allows us to view the optimisation of a pseudo-Boolean function as a continuous nonlinear optimisation problem over the unit hypercube $\mathbb{U}^n$:

Theorem 3.3 ([40]). *For any pseudo-Boolean function f,*

$$\min_{\mathbf{x} \in \mathbb{B}^n} f(\mathbf{x}) = \min_{a \in \mathbb{U}^n} f(a).$$

By means of the *derivative* of a pseudo-Boolean function:

$$\begin{aligned}\Delta_i(\mathbf{x}) &\overset{\text{def}}{=} \frac{\partial f}{\partial x_i}(\mathbf{x}) \\ &= f(x_1, \ldots, x_i = 1, \ldots, x_n) - f(x_1, \ldots, x_i = 0, \ldots, x_n),\end{aligned} \tag{3.4}$$

a simple characterisation of local optima can be given:

Theorem 3.4 ([40])**.** *Let $f : \mathbb{B}^n \to \mathbb{R}$ be a pseudo-Boolean function. The vector $\mathbf{x}$ is a local minimum of f if and only if*

$$x_i = \begin{cases} 1 & \textit{if } \Delta_i(\mathbf{x}) < 0, \\ 0 & \textit{if } \Delta_i(\mathbf{x}) > 0 \end{cases} \tag{3.5}$$

for all $i = 1, \dots, n$.

This criterion can be used to obtain a characterisation of the i-th component of all local minima in terms of the other components of the vector. If this is successively done for all components in an elimination-like scheme, it results in a global minimum. If all successive derivatives of f and its transformations during the elimination process depend only on some bounded number $n' < n$ of variables, this algorithm can be shown to run in polynomial time of the input size and $2^{n'}$ (see [51]). However, in general the execution time is exponential in the input size.

An alternative practical approach is to exploit the fact that the optimisation of a pseudo-Boolean function can be reduced in polynomial time to the optimisation of a quadratic pseudo-Boolean function [148] and design heuristics for the minimisation of this subtype of pseudo-Boolean functions.

One widely employed heuristic based on this fact is the DDT heuristic ("devour, digest, tidy up"), which is a greedy algorithm successively restricting the Boolean cube to smaller subcubes in which the terms with the highest remaining coefficient vanish [40]. It is described in Algorithm 3.1.

Algorithm 3.1 The DDT heuristic for pseudo-Boolean optimisation.

```
Input: Quadratic posiform $\phi(\mathbf{x}) = \sum_{T \subset \mathbf{L}} a_T \prod_{u \in T} u$.
Output: Approximation $\tilde{\mathbf{x}}$ of a minimum of $\phi$.
1: $\mathcal{S} \leftarrow \emptyset$, $\tilde{\phi} \leftarrow \phi$
2: [Devour] Find term $T$ with largest coefficient $a_T$ and set $\mathcal{S} = \mathcal{S} \cup \{T\}$.
3: [Digest] Draw all logical conclusions $\mathcal{C}$ from the Boolean equation
       $$\bigvee_{T \in \mathcal{S}} T(\mathbf{x}) = 0. \tag{3.6}$$
4: [Tidy up] Substitute the consequences $\mathcal{C}$ into $\tilde{\phi}$ and simplify the result.
5: if $\tilde{\phi} \equiv$ const then
6:     return a solution $\tilde{\mathbf{x}}$ of equation (3.6).
7: else
8:     go to step 2.
9: end if
```

Due to the special form of the quadratic Boolean equation (3.6), the conclusions can be determined in polynomial time [40]. However, while reported to work well in practice, this heuristic does not provide strong guarantees about the quality of the solution it determines. In particular, we show that it does not always approximate the optimum by a constant factor.

Definition 3.5. Let $\mathcal{P}$ a minimisation problem and denote by $\mathcal{P}(i)$ the optimum of problem instance i . A heuristic $\mathcal{H}$ for $\mathcal{P}$ is called a *constant-factor approximation algorithm for* $\mathcal{P}$ if there exists an $\varepsilon \geq 0$ such that $\frac{\mathcal{H}(i)}{\mathcal{P}(i)} \leq 1 + \varepsilon$ for all problem instances i.

Theorem 3.6. *The DDT heuristic is not a constant-factor approximation algorithm for the pseudo-Boolean minimisation problem* $\mathcal{PBO}$*.*

Proof. We will prove the statement by explicitly constructing a family of problem instances for which the DDT heuristic fails to provide a constant-factor approximation. Specifically, for any dimension $n \geq 3$, define the following problem instance in the variables $x_1, \ldots, x_n$:

$$f(x_1, \ldots, x_n) = (a+1)\overline{x_1} + \sum_{1<i\leq n} a\overline{x_i} + \sum_{1<j\leq n} bx_1x_j \tag{3.7}$$

for positive integers a, b satisfying the conditions

$$a > b \tag{3.8}$$

$$\text{and} \qquad a + 1 < (n-1)b. \tag{3.9}$$

Note that inequalities (3.8) and (3.9) are not contradictory, for example, $a = 4, b = 3$ is a valid choice for all $n \geq 3$.

The DDT heuristic, applied to this posiform, will take the following steps:

1. Select literal $\overline{x_1}$, set $\mathcal{C} := \{x_1 = 1\}$ and $\tilde{\phi} := \sum_{1<i\leq n} a\overline{x_i} + \sum_{1<j\leq n} bx_j$.
2. This step is repeated $n-2$ times for $k = 2, \ldots, n-1$: Select literal x_k, set $\mathcal{C} := \{x_1 = 1, \ldots, x_k = 1\}$ and $\tilde{\phi} := \sum_{k<i\leq n} a\overline{x_i} + \sum_{k<j\leq n} bx_j + (k-1)b$.
3. Now, $\tilde{\phi}$ is simplified to $a\overline{x_n} + bx_n + (n-2)b$. Select literal x_n, set $\mathcal{C} := \{x_1 = \cdots = x_n = 1\}$ and $\tilde{\phi} := (n-1)b$, which is a constant expression in the x_i.
4. Output the solution $\mathbf{x} = (1, \ldots, 1)$ with $f(\mathbf{x}) = (n-1)b$.

However, f does not have a minimum at $\mathbf{x} = (1, \ldots, 1)$ for any $n \geq 3$. To see this, note that expression (3.7) can be characterised in terms of the Hamming weight $w_h(\cdot)$ of the vector $(x_2, \ldots, x_n)$: We have

$$f(x_1, \ldots, x_n) = \begin{cases} (a+1) + (n-1-w_h(x_2, \ldots, x_n)) \cdot a & \text{if } x_1 = 0, \\ (n-1-w_h(x_2, \ldots, x_n)) \cdot a + w_h(x_2, \ldots, x_n) \cdot b & \text{if } x_1 = 1. \end{cases} \tag{3.10}$$

Suppose first that $x_1 = 0$. Abbreviating the weight of $(x_2, \ldots, x_n)$ by w, equation (3.10) becomes $g(w) = a + 1 + (n-1-w)a$ and since

$$\min_{(x_2, \ldots, x_n) \in \mathbb{B}^{n-1}} f(0, x_2, \ldots, x_n) = \min_{w \in \{0, \ldots, n-1\}} g(w),$$

the minimum of $f(\mathbf{x})\big|_{x_1=0}$ is given by

$$\begin{aligned} \arg\min_{w \in \{0, \ldots, n-1\}} g(w) &= \arg\min_{w \in \{0, \ldots, n-1\}} -aw + na + 1 \\ &= n-1, \end{aligned}$$

and we have $g(n-1) = f(0, 1, \ldots, 1) = a + 1$ for all n.

If, on the other hand, $x_1 = 1$, then equation (3.10) becomes $h(w) = (n-1-w)a + wb$ and the minimum of $f(\mathbf{x})\big|_{x_1=1}$ is given by

$$\begin{aligned} \arg\min_{w \in \{1, \ldots, n-1\}} h(w) &= \arg\min_{w \in \{1, \ldots, n-1\}} (b-a)w + (n-1)a \\ &= n-1, \end{aligned}$$

since $b - a < 0$ because of condition (3.8). We have $h(n-1) = f(1, 1, \ldots, 1) = (n-1)b$, which is strictly greater than $a+1$ per assumption (3.9). Consequently, f attains its minimum value $a+1$ at $\mathbf{x} = (0, 1, \ldots, 1)$. Furthermore, this minimum is unique since there is only one binary vector of length $n-1$ with Hamming weight $n-1$.

Comparing the output of the DDT heuristic with the actual minimum, we find that

$$\frac{\mathcal{DDT}(f)}{\mathcal{PBO}(f)} = \frac{b(n-1)}{a+1},$$

which is not a constant since it still depends on the problem size. This proves the claim. □

This result by itself does not imply that the DDT heuristic cannot be useful in some practical applications. As we will see, however, it constitutes a significant theoretical disadvantage in comparison to nonsmooth algorithms. Experimental results for the application of the DDT heuristic to pseudo-Boolean models of state and key recovery for the stream cipher MICKEY are given in Section 3.4.6.

3.3 Nonsmooth cryptanalysis

Approaches to converting systems of equations corresponding to cryptanalytic problems such as recovering a previous internal state or the key to the reals and subjecting them to numerical optimisation have so far been focusing on polynomial representations [39, 127]. The success of this approach has been shown to significantly rely on both the overall dimension and the individual degree of the expressions not becoming prohibitively high [37, 127]. While degree can be traded for dimensionality and vice versa, especially the models for full-scale algorithms turned out to be too big for most solvers.

3.3.1 A nonsmooth model of Boolean equations

An alternative approach to reducing the problems associated with the degree is to choose another real-valued representation that inherently avoids higher degrees. We propose the following representation providing this property:

Definition 3.7 (Nonsmooth representation and conversion)**.** Consider arbitrary expressions T over $\mathbb{F}_2$ using the operations $\bar{\cdot}$ (negation), $\wedge$ (logical AND), $\vee$ (logical OR) and $\oplus$ (EXOR). Their conversion $\psi(T)$ into nonsmooth representation over the reals is recursively defined as

$$\psi(T) = \begin{cases} 0 \in \mathbb{R}, & \text{if } T = 0 \in \mathbb{F}_2, \quad (3.11) \\ 1 \in \mathbb{R}, & \text{if } T = 1 \in \mathbb{F}_2, \quad (3.12) \\ 1 - \psi(a), & \text{if } T = \overline{a}, \quad (3.13) \\ \min\{\psi(a), \psi(b)\}, & \text{if } T = a \wedge b, \quad (3.14) \\ \max\{\psi(a), \psi(b)\}, & \text{if } T = a \vee b, \quad (3.15) \\ \max\{\psi(a), \psi(b)\} - \min\{\psi(a), \psi(b)\} & \\ \qquad = |\psi(a) - \psi(b)|. & \text{if } T = a \oplus b. \quad (3.16) \end{cases}$$

It is easy to see that for $a, b \in \mathbb{B}$, the roots of the Boolean expressions correspond to the roots of their real-valued counterparts. However, while this correspondence also holds vice versa in the case of $\bar{\cdot}$, $\wedge$ and $\vee$ (e.g. $\min\{a, b\} = 0$ if and only if a or b are zero), $|a - b| = 0$ holds for any $a = b \in \mathbb{R}$, so specifically also for all values in the unit interval. This implies that while any solution in the Boolean domain is also a solution over the reals, there does not necessarily have to be a Boolean counterpart to all solutions of the real-valued model.

We refer to the maximum number of nested nonsmooth operations $\min, \max$ and $|\cdot|$ in an expression as its "nonsmooth degree", or just "degree" if the context is clear:

Definition 3.8. Consider the nonsmooth representation $T_N = \psi(T)$ of a Boolean expression T over variables $x_1, \ldots, x_m$ as in Definition 3.7. The *nonsmooth degree* $d(T_N)$ of T_N is recursively given by

$$d(T_N) = \begin{cases} 1 + \max\{d(a), d(b)\}, & \text{if } T_N = \min\{a, b\} \text{ or } T_N = \max\{a, b\} \\ & \quad \text{or } T_N = |a - b|, \\ d(a) & \text{if } \ T_N = 1 - a, \\ 0 & \text{if } \ T_N = x_i, 0, 1. \end{cases}$$

For example, the nonsmooth degree of the expression x is zero, and $\max\{1 - x, |x - y|\}$ has degree two.

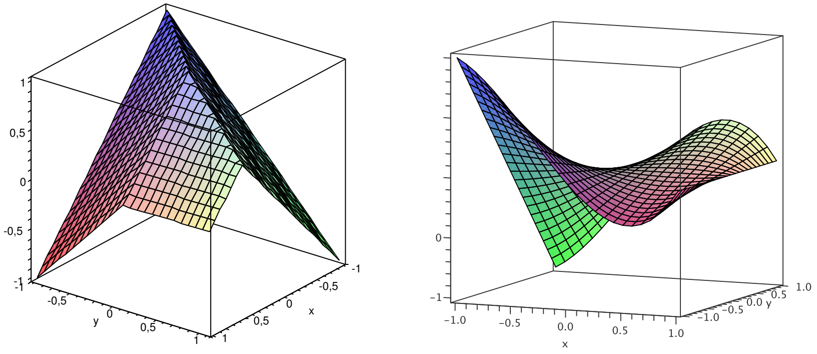
A visual depiction of the nonsmooth function $F_N(x, y) = 1 - \max\{x, |x - y|\}$ resulting from the conversion of the Boolean expression $F(x, y) = x \vee (x \oplus y)$ can be found in Figure 3.1, along with the polynomial conversion of $F(x, y)$ according to the standard representation method of Definition 3.1. In order to provide a more complete picture, the unit hypercube has been extended to $[-1, 1]^2$ in this figure.

Definition 3.9 (Polytope)**.** For fixed $0 \neq a \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$, we denote the closed half-space defined by a and α by

$$\mathcal{H}_{a,\alpha} \overset{\text{def}}{=} \{x \in \mathbb{R}^n \mid a^T x \geq \alpha\}.$$

A set $\mathcal{P} \subseteq \mathbb{R}^n$ is called a *convex polytope* (or *polytope* for short) if it can be written as the intersection of a finite number of closed half-spaces.

As can be seen in Figure 3.1a, the resulting expressions are actually piecewise linear on different subsets partitioning the whole $\mathbb{R}^n$. However, the objective function is usually not linear due to the use of a norm (3.1). When used in an



(a) Nonsmooth conversion.

(b) Polynomial conversion (standard representation).

Figure 3.1: Conversion of the Boolean function $F(x, y) = \overline{x \vee (x \oplus y)}$ to the reals.

equation-solving (instead of minimisation) context, the problem of obtaining a solution could be rewritten as a feasibility problem, that is, as determining an element of a complex intersection of cutting hyperplanes in $\mathbb{R}^n$. This seems to suggest that standard linear programming techniques could be used to solve this problem. However, linear programming requires the optimum to be in a vertex of the resulting polytope, which need not to be the case for our model. Therefore, linear programming and related techniques are not applicable in the nonsmooth model.

It is also worth noting that in most applications of nonsmooth optimisation, the nonsmoothness arises as an inherent property of the problem that is difficult to overcome. By contrast, in our case, we deliberately choose a nonsmooth representation for a discrete problem.

3.3.2 Minimising nonsmooth functions: The Bundle method

Naturally, this nonsmooth representation is unsuitable for most numerical solvers, since they expect differentiable functions to calculate descent directions. However, nonsmooth analysis is a branch of calculus dealing with this type of functions [64]. The crucial observation is that those functions and compositions of them are still locally Lipschitz-continuous:

Definition 3.10 (Local Lipschitz continuity)**.** A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is called *locally Lipschitz-continuous* in a point x if there exist $L > 0$ and $\delta > 0$ such that

$$\|f(y) - f(x)\| \leq L\|y - x\| \qquad \forall y \in \mathcal{B}_\delta(x),$$

where $\mathcal{B}_\delta(x) := \{y \in \mathbb{R}^n : \|y - x\| < \delta\}$ denotes the open ball of radius δ centered at x.

Informally, this means that the slope of all secants of the function's graph in some neighbourhood of x is bounded by the *Lipschitz constant* L (which may depend on x); f cannot change arbitrarily fast around x. This property enables local line-search based techniques.

Convex functions are a particularly convenient subclass of locally Lipschitz-continuous functions:

Definition 3.11 (Convex function)**.** A function $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$ is *convex* in X if and only if X is a convex set and for all $x, y \in X$ and all $\lambda \in [0, 1]$,

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y).$$

Intuitively, convex functions have the property that any line segment connecting two points on its graph lies on or above the graph. It follows from the definition that a local minimum of a convex function is automatically also a global minimum.

For the sake of a clearer exposition, we will first deal with the convex case, but the developed concepts readily carry over to the Lipschitzian case. The only major differences concern guarantees regarding global optimisation and, sometimes, the rate of convergence. An outline of how to adapt the algorithms to the nonconvex case is given in Section 3.3.3.

A second prerequisite for a nonsmooth optimisation algorithm is a characterisation of descent directions.

Definition 3.12 (Direction of descent)**.** Let $f : \mathbb{R}^n \to \mathbb{R}$. A direction $s \in \mathbb{R}^n \setminus \{0\}$ is called a *direction of descent* (or *descent direction*) of f at point x if the directional derivative of f in x exists and is negative:

$$f'(x, s) := \lim_{\lambda \to 0+} \frac{f(x + \lambda s) - f(x)}{\lambda} < 0.$$

A *steepest descent direction* of f at point x is a direction s such that

$$f'(x, \frac{s}{\|s\|}) = \min_{\|t\|=1} f'(x, t).$$

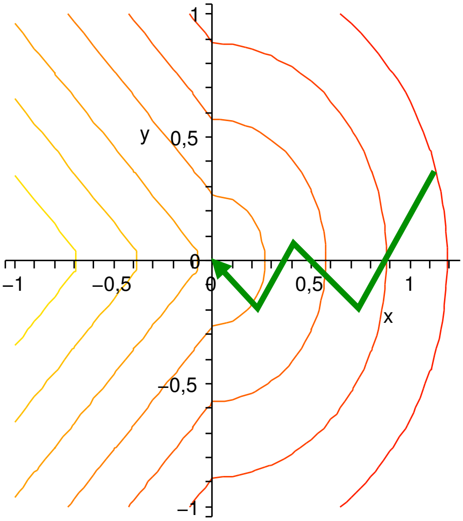


Figure 3.2: Example illustrating the inapplicability of gradient descent search to nonsmooth objective functions.

If f is differentiable in x, then

$$s := -\lambda \cdot \nabla f(x)$$

gives a direction of steepest descent.

Since the expressions arising from equations (3.11)–(3.16) are only nondifferentiable at a finite number of points, the computation of directional derivatives (and therefore descent directions) is possible in most cases. A straightforward application of a line-search strategy to find steepest directions of descent is however not guaranteed to succeed. One can construct functions (see e.g. [47]) which are only nonsmooth at one point and still cause the gradient descent method to fail. A contour map of such an example is depicted in Figure 3.2, where the objective function is smooth on $\mathbb{R}^2 \setminus \{(0,0)\}$. The sequence of steepest descent directions (depicted in green) converges to the nonsmooth point $(0,0)$, without taking into account that $(-\sigma, 0)$ still is a direction of steepest descent in the origin.

In nonsmooth analysis, the role of the gradient is replaced by the notion of *subgradients*:

Definition 3.13 (Subgradient, subdifferential)**.** Let $X \subseteq \mathbb{R}^n$ and $f : X \to \mathbb{R}$ convex. The vector $g \in \mathbb{R}^n$ is called a *subgradient* of f in x if

$$f(y) - f(x) \geq g^T(y - x) \qquad \forall y \in X.$$

The *subdifferential* $\partial f(x)$ is defined as the set of all subgradients of f at point x.

However, subgradients cannot directly be used to calculate descent directions. In contrast to the smooth case, $-g$ for $g \in \partial f(x)$ is not necessarily a direction of descent.

An intuitive way of using the subdifferential for optimisation is the following: Consider the convex minimisation problem $\min_{x \in X} f(x)$ with $X \subset \mathbb{R}^n$ a compact set. Given pairs (x^k, g^k) with $x^k \in X$ and $g^k \in \partial f(x^k)$, observe that the linear mapping

$$l_k(x) := f(x^k) + {g^k}^T (x - x^k) \tag{3.17}$$

is a minorant of f for each k. Consequently, an underestimation of f is given by the piecewise linear model

$$F_k^{\mathrm{cp}}(x) := \max_{0 \le j \le k} l_j(x), \tag{3.18}$$

called the *cutting plane* model of f. By determining the next point of iteration x^{k+1} as the minimum of $F_k^{\mathrm{cp}}(x)$ subject to $x \in X$, this model is incrementally refined by inclusion of another linear minorant l_{k+1} given by the subgradient g^{k+1} at x^{k+1}. The gap between f and l_k is called the *linearisation error* and denoted α_k. By the extreme value and Heine-Borel theorems [162], the minimisation of $F_k^{\mathrm{cp}}(x)$ is well-defined. This idea is illustrated in Figure 3.3 (note that in this figure, the bundle has already been pruned, so that y_1, y_2 and y_3 do not correspond to three successive iterations).

There are two main issues with this method. First, it only applies to compact sets X which means that it cannot be used for unconstrained minimisation. Second, it can require a very large number of iterations which also causes an intolerable growth of the size of the cutting plane model. An example of this is provided in [87], where a minimisation problem over the unit ball in only 25 variables with nonsmooth degree 2 is shown to require at least 2^{59} iterations of the cutting plane method (and, accordingly, a model of 2^{59} planes).

The first issue can be resolved by adding a quadratic regularisation term and determining the next point of iteration by minimising $F_k^{\mathrm{cp}}(x) + \frac{1}{\gamma_k} \|x - x^k\|^2$ instead, where $\gamma_k > 0$ is a parameter that can be adaptively chosen during the iteration. The second issue can be addressed by removing old and potentially irrelevant parts of the cutting plane model. This leads to the so-called *bundle* idea [126].

The idea of the Bundle method can be interpreted in two alternative ways: First, as a refinement of the cutting plane method which dynamically adapts

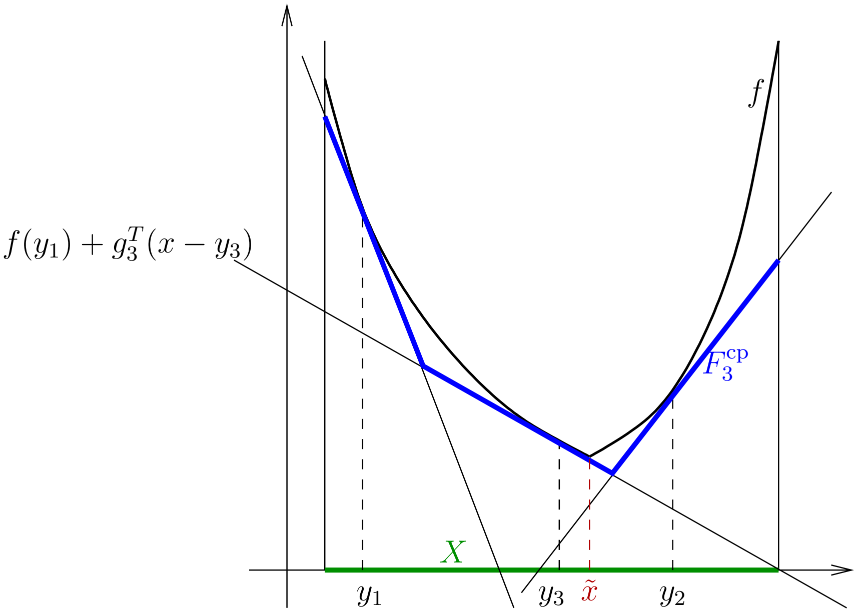


Figure 3.3: The idea of the Bundle method: approximate f from below by a cutting plane model. $\tilde{x}$ denotes the minimum of f on X.

the model to the progress of the iteration: If the improvement suggested by the next step is too small, a "null step" is made to improve the model by sampling more subgradient information before proceeding; otherwise, a "serious step" is performed in which old and potentially irrelevant parts of the cutting plane model are removed. This ensures that the size of the model does not increase arbitrarily.

Its second – dual – interpretation, views the set of all convex combinations of the subgradients in the bundle as an approximation of the so-called ε-subdifferential:

Definition 3.14 (ε-subgradient, ε-subdifferential)**.** Let $f : \mathbb{R}^n \to \mathbb{R}$ convex and $\varepsilon \geq 0$. A vector $g \in \mathbb{R}^n$ is called *ε-subgradient* of f in x if

$$f(y) - f(x) \geq g^T(y - x) - \varepsilon \qquad \forall y \in \mathbb{R}^n.$$

The *ε-subdifferential* $\partial_\varepsilon f(x)$ is the set of all subgradients of f in x.

The advantage of the ε-subdifferential in a point x_0 is that it contains information about subdifferentials in the neighbourhood of x_0 in the sense that for any $\varepsilon > 0$, there exists a $\delta > 0$ such that

$$\bigcup_{y \in \mathcal{B}_\delta(x_0)} \partial f(y) \subset \partial_\varepsilon f(x_0).$$

Furthermore, it is easy to see that stationary points with respect to the ε-subdifferential are necessarily ε-optimal:

Proposition 3.15. *For a convex function $f : \mathbb{R}^n \to \mathbb{R}$ and $\varepsilon \geq 0$, the following statements are equivalent:*

(1) $\bar{x}$ is ε-optimal, i.e. $f(\bar{x}) \leq f(x) + \varepsilon$ for all x.

(2) $0 \in \partial_\varepsilon f(\bar{x})$.

Let $P_X : x \mapsto \arg\min_{y \in X} \|y - x\|$ denote the orthogonal projection onto a convex closed set $X \subseteq \mathbb{R}^n$. It can be shown that the negative of the orthogonal projection of the origin onto $\partial_\varepsilon f(x)$ is a direction of steepest descent for all non-ε-optimal points [87]. The main problem with exploiting this, however, is that $\partial_\varepsilon f(x)$ as a whole (in contrast to finding a single subgradient) and therefore also $-P_{\partial_\varepsilon f(x)}(0)$ are difficult to determine. The Bundle method therefore approximates $\partial_\varepsilon f(x)$ by the convex hull of a bundle of subgradients to determine directions of descent, see Figure 3.4.

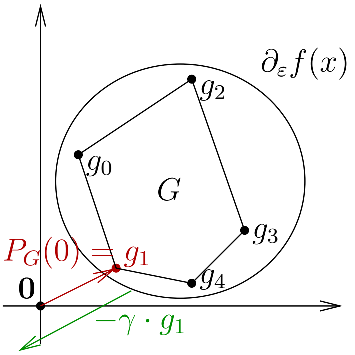


Figure 3.4: Determining descent directions via projection on the convex approximation of the ε-subdifferential. $G = \operatorname{conv}\{g^0, \ldots, g^k\}$ denotes the convex hull of the subgradients in the bundle.

A simplified version [157] of the Bundle method for the convex real-valued case is described in Algorithm 3.2. The parameter γ combines the Bundle method with ideas from trust-region algorithms [151] by specifying a trust region radius for the cutting plane subproblem (3.19). The version presented here does put relatively little effort into fine-tuning the trust region. In [151], for instance, an extensive inner iteration is performed to assess the trust region. The parameter $\eta \in (0, 1)$ specifies how big the improvement of the objective function compared to the improvement predicted by the cutting plane model has to be in order to carry out a serious step. Finally, ε specifies an acceptable (additive) defect for the approximation of the minimum.

We briefly outline the operation of Algorithm 3.2. Throughout the description, vectors are indicated with Latin, and real numbers with Greek characters. The algorithm keeps a bundle of points y^j and corresponding subgradients g^j for $j \in J_k$. In each step, first the trust region radius γ^k is assessed and a new step vector s is determined by solving subproblem (3.19). This subproblem is a simple reformulation of the nonsmooth problem of minimising (3.18): Instead of minimising a piecewise linear max-type function, we find a simultaneous minimum upper bound for all the linear functions. Due to the regularisation term $\|s\|^2$, this is a quadratic optimisation problem with linear inequality constraints, which can be efficiently solved, for instance with the active set strategy [131].

Then, the stop criterion is evaluated: If the previous step and the sum of the linearisation errors with respect to the cutting planes with active Lagrange multipliers ($\lambda_j^k > 0$) was small, we have reached an ε-optimal point and stop. If this is not the case, the algorithm compares the actual improvement of the objective function to the prediction by the cutting plane model. If the improvement is significant enough, the iteration moves forward and prunes the bundle by only keeping those points y^j and subgradients g^j with nonzero Lagrange multipliers (i.e. those cutting planes that support f from below). Otherwise, more subgradient information is sampled around the current point of iteration.

For convex objective functions, this method converges globally. If f is not convex, a few modifications to the algorithm are necessary, and additional assumptions (i.e. requiring f to be *semi-smooth*) are required for a global convergence result [151]. This is detailed in the following section.

3.3.3 Dealing with nonconvex objective functions

The main advantage in convex (and dually, concave) optimisation is that local optima are global optima by definition. If the objective function is merely locally Lipschitz-continuous, a more general notion of subdifferential is required. While there exist many proposals for subdifferentials of locally

Algorithm 3.2 The Bundle method for nonsmooth optimisation (convex case).

```
Input: $f : \mathbb{R}^n \to \mathbb{R}$ convex.
Output: Approximation $\tilde{x}$ of a minimum of $f$.
Choose starting point $x^0$ and parameters $\gamma^+ \geq \gamma^- > 0$, $\eta \in (0,1)$ and $\varepsilon \geq 0$.
Determine $g^0 \in \partial f(x^0)$ and set $y^0 \leftarrow x^0$, $\alpha_0 \leftarrow 0$, $J_0 \leftarrow \{0\}$.
for $k = 0, 1, \ldots$ do
    Choose $\gamma_k \in [\gamma^-, \gamma^+]$ and determine a KKT tuple $(s^k, \xi^k, \lambda^k)$ of the problem
        $$\begin{aligned} \min_{s,\xi} \quad & \xi + \frac{1}{2\gamma_k}\|s\|^2 \\ \text{s.t.} \quad & {g^j}^T s - \alpha_j^k \leq \xi \quad (\forall j \in J_k) \end{aligned} \qquad (3.19)$$
    Set $v^k \leftarrow -\frac{1}{\gamma^k} s^k$, $\varepsilon_k \leftarrow \sum_{j \in J_k} \lambda_j^k \alpha_j^k$.
    if $\|v^k\| \leq \varepsilon$ and $\varepsilon_k \leq \varepsilon$ then
        return $\tilde{x} \leftarrow x^k$
    end if
    if $f(x^k + s^k) - f(x^k) \leq \eta \xi^k$ then    ▷ serious step
        Set
            $y^{k+1} \leftarrow x^k + s^k$
            $x^{k+1} \leftarrow x^k + s^k$
            $J_{k+1} \leftarrow \{j \in J_k \mid \lambda_i^k > 0\} \cup \{k+1\}$
    else    ▷ null step
        Set
            $y^{k+1} \leftarrow x^k + s^k$
            $x^{k+1} \leftarrow x^k$
            $J_{k+1} \leftarrow \{j \in J_k \mid \lambda_i^k > 0\} \cup \{j \mid y^j = x^k\} \cup \{k+1\}$
    end if
    Calculate
        $f^{k+1} \leftarrow f(y^{k+1})$
        $g^{k+1} \in \partial f(y^{k+1})$
        $\alpha_j^{k+1} \leftarrow f(x^k) - l_j(x^k) \quad \forall j \in J_{k+1}$.
end for
```

Lipschitz-continuous functions, the Clarke subdifferential [47] is most widely used:

Definition 3.16 (Clarke subdifferential)**.** Let $f : X \subset \mathbb{R}^n \to \mathbb{R}$ be locally Lipschitz-continuous and denote by $X_d \subset X$ the set of points where f is differentiable. The *Clarke subdifferential* of f at point $x \in X$ is the set

$$\begin{aligned} \partial^{\mathrm{cl}} f(x) := \operatorname{conv}\{v \in \mathbb{R}^n \mid \exists\ (x_k) \text{ with } x_k \in X_d \text{ and} \\ x_k \to x, \nabla f(x_k) \to v \text{ for } k \to \infty\}. \end{aligned}$$

For instance, the Clarke subdifferential of the absolute value function $f(x) = |x|$ at $x = 0$ is $\partial^{\mathrm{cl}} f(0) = \operatorname{conv}\{-1, 1\} = [-1, 1]$. Another commonly used notion of subdifferential is the Demyanov-Rubinov quasidifferential [65].

Unlike the convex case, $0 \in \partial^{\mathrm{cl}} f(\bar{x})$ is only a *necessary*, not a sufficient condition for $\bar{x}$ to be a local or global minimizer of f. The same holds for maximizers. Consequently, the algorithms can only search for stationary points that are not guaranteed to be even locally optimal.

When applying the bundle idea to nonconvex nonsmooth optimisation, one major issue arises: The cutting plane model (3.18) is not any more an underestimate for f, the linearisation error α_j^k can become very small or even negative, which is especially problematic for very distant trial points y^j because the corresponding subgradients do not improve the model. A possible solution to this problem is to replace the linearisation error α_j^k by the so-called *subgradient locality measure* [151]:

$$\beta_j^k := \max\{a_j^k, \mu \|x^k - y^j\|^2\} \tag{3.20}$$

with $\mu \geq 0$ parametrising the distance measure. Alternatively, the cutting plane model can be split in an under- and an overestimation of f. The next point in the bundle is then selected in a way that both under- and overestimates predict a significant improvement. This has recently been proposed by [73].

Due to Rademacher's famous theorem [18], a locally Lipschitz-continuous function f is differentiable almost everywhere. This implies that the set of points where f is not smooth has (Lebesgue) measure zero. If, additionally, $\partial^{\mathrm{cl}} f(x)$ is a point-based approximation of f, that is, if

$$\sup_{v \in \partial^{\mathrm{cl}} f(x+s)} \|f(x+s) - f(x) - v^T s\| = o(\|s\|) \qquad \text{for } s \to 0,$$

f is called *semi-smooth* and the rate of convergence is comparable to the convex case [18, 151]. It is worth noting that this only applies to the rate of convergence towards a *stationary point* which does not necessarily have to be a global minimum.

The expressions used in our nonsmooth model of Boolean equations given by equations (3.11)–(3.16) and arbitrary compositions of them are semi-smooth.

3.4 Applying techniques from nonsmooth optimisation to MICKEY

3.4.1 The stream cipher MICKEY

The stream cipher MICKEY is a hardware-oriented stream cipher designed by Babbage and Dodd with a security level of 80 bits [4]. It was submitted to the eSTREAM stream cipher competition and selected as one of three hardware-oriented algorithms in the final portfolio. While its first version could be attacked by a time-memory tradeoff with online complexity smaller than exhaustive key search (albeit more expensive precomputation, see [88]), no cryptanalysis results have been published for the current version 2.0.

MICKEY consists of two registers R and S of 100 bits each, R being a linear feedback shift register (LFSR) with maximum period, and S being a nonlinear feedback shift register. A cell $s_i^{(t)}$ of S at time t is updated by the rule

$$s_i^{(t+1)} = s_{i-1}^{(t)} \oplus \left(f^{(t)} \wedge F_i^{c_S}\right) \oplus \left(\left(C_i^0 \oplus s_i^{(t)}\right) \wedge \left(C_i^1 \oplus s_{i+1}^{(t)}\right)\right),$$

where $f^{(t)}$ is the feedback bit, $(F^{0,1})_i$ are constant binary sequences, and the complementation of s_i and s_{i+1} depends on the value of the constant binary sequences $(C^0)_i$ and $(C^1)_i$, respectively.

The clocking of both registers is done in a way that depends on bits from both registers. More specifically, if the control bit $c_R := s_{34} \oplus r_{67}$ is equal to one, the linear register R is clocked $2^{50} - 157$ times ahead by XOR-ing each bit back into the current stage while shifting. Otherwise, R is clocked normally. Analogously, the control bit $c_S := s_{67} \oplus r_{33}$ determines whether the sequence F^0 or F^1 is used in the update of S.

During the setup phase, R and S are initialised with all zeros, afterwards the IV and the key are clocked in bit by bit, followed by 100 steps of preclocking with the input bit set to zero. The input procedure consists of XOR-ing the input bit to the feedback bits of both registers. Additionally, the cell s_{50} of the S register is XOR-ed into the feedback of the R register. After the setup phase, keystream bits are generated by first outputting the XOR of the leftmost bits s_0 and r_0 of both registers and then clocking both registers (without the mixing of s_{50} into the feedback of R).

3.4.2 A scalable variant

For cryptanalysis purposes, it is often useful to work with reduced or weakened versions of the algorithm in consideration. This can also provide a useful indication about the security margin of the full algorithm. Besides the obvious way of weakening MICKEY by reducing the number of steps during the preclocking, we have defined a scalable variant of MICKEY called *Mini-MICKEY* where both the key length k and the register size n can be varied independently. The structure of Mini-MICKEY has been chosen to resemble that of MICKEY as closely as possible.

The size of each register of Mini-MICKEY can be varied according to $4 \leq n \leq 100$; for the key length we require $4 \leq k \leq \min\{80, 2n\}$ bits, but k can be arbitrary in principle. Mini-MICKEY's S register is a truncated version of MICKEY's S register, that is, the sequences $(C^0)_i, (C^1)_i$ and $(F^0)_i, (F^1)_i$ are simply restricted to their first n elements. The LFSR R is defined by the first primitive polynomial of degree n in Table D in chapter 10 of [111], with one exception: if $n = 100$, MICKEY's original feedback polynomial is used. The control bits for the irregular mutual clocking are defined by

$$c_R := s_{\lfloor n/3 \rfloor} \oplus r_{\lfloor 2n/3 \rfloor} \qquad \text{and} \tag{3.21}$$

$$c_S := s_{\lfloor 2n/3 \rfloor} \oplus r_{\lfloor n/3 \rfloor}. \tag{3.22}$$

As in MICKEY, keystream bits are generated by the output function $s_0 \oplus r_0$. The mixing during the setup phase is done by XOR-ing $s_{\lfloor n/2 \rfloor}$ into the feedback bit of R. After the loading of the IV and the key, a preclocking of n steps is applied. A general overview of the structure of Mini-MICKEY is given in Figure 3.5.

It should be noted that this does not exactly yield the full MICKEY when parameters $k = 80$ and $n = 100$ are selected. The differences lie in the selection of indices for the control bits: In (3.21) and (3.22), $s_{\lceil n/3 \rceil}, s_{\lceil 2n/3 \rceil}$ and $r_{\lceil 2n/3 \rceil}$ would have to be used instead of their floor function variants to match the indices used in the full MICKEY. However, the use of the ceiling function would imply that in the important small cases $n = 4, 5$, the control bits would depend on the first and the feedback bit, in contrast to MICKEY's design choice to determine the control bits as the XOR of "inner" register cells. On the other hand, we verified that selecting bits 33 and 66 instead of 34 and 67 in

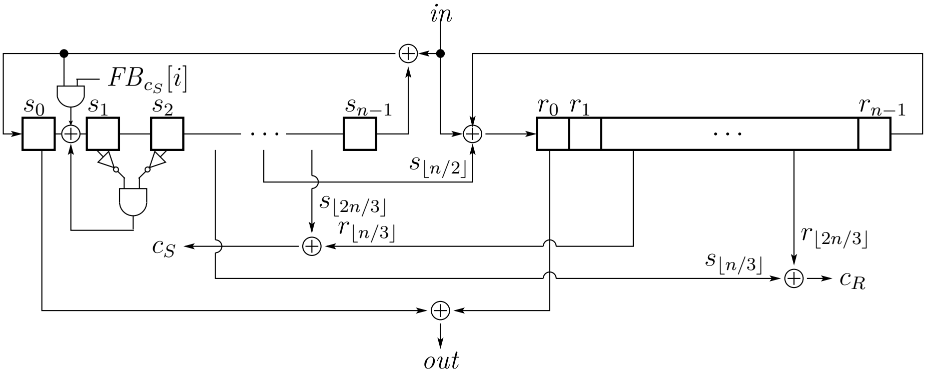

in
$FB_{c_S}[i]$
s_0
s_1
s_2
s_{n-1}
r_0
r_1
r_{n-1}
$s_{\lfloor n/2 \rfloor}$
$s_{\lfloor 2n/3 \rfloor}$
$r_{\lfloor n/3 \rfloor}$
c_S
$r_{\lfloor 2n/3 \rfloor}$
$s_{\lfloor n/3 \rfloor}$
c_R
out

Figure 3.5: Architectural overview of Mini-MICKEY.

this variant with $n = 100$ does not cause Mini-MICKEY to behave numerically different than the original MICKEY.

3.4.3 Equations representing Mini-MICKEY

The state update in both of MICKEY's registers is usually dense in the sense that in each clocking step, potentially all bits of both registers can require an update. In this case, the LFSR performs the "jumping" operation described in Section 3.4.1. Due to the mutual clocking, this occurs depending on bits from both the linear and the nonlinear register. Consequently, about $2nc$ new equations are generated for c clockings.

As an illustration, using the standard nonsmooth representation according to equations (3.11)–(3.16), the following equations are generated for Mini-MICKEY with $n = k = 4$:

First clock:

$$r^{(1)}_{0,1,3} = k_0$$

$$r_2 = 0$$

$$s^{(1)}_{0,1,2,3} = k_0$$

i-th clock ($i > 1$):

$$
\begin{aligned}
c_R^{(i)} &= \left| s_1^{(i)} - r_2^{(i)} \right| \\
c_S^{(i)} &= \left| s_2^{(i)} - r_1^{(i)} \right| \\
r_0^{(i+1)} &= \left| \left| r_3^{(i)} - \left| k_i - s_2^{(i)} \right| \right| - \min\left\{ c_R^{(i)}, r_0^{(i)} \right\} \right| \\
r_1^{(i+1)} &= \left| \left| r_0^{(i)} - \left| r_3^{(i)} - \left| k_i - s_2^{(i)} \right| \right| \right| - \min\left\{ c_R^{(i)}, r_1^{(i)} \right\} \right| \\
r_2^{(i+1)} &= \left| r_1^{(i)} - \min\left\{ c_R^{(i)}, r_2^{(i)} \right\} \right| \\
r_3^{(i+1)} &= \left| \left| r_2^{(i)} - \left| r_3^{(i)} - \left| k_i - s_2^{(i)} \right| \right| \right| - \min\left\{ c_R^{(i)}, r_3^{(i)} \right\} \right| \\
s_0^{(i+1)} &= \min\left\{ \left| s_3^{(i)} - k_i \right|, \max\left\{ \min\left\{ c_S^{(i)}, F_0^1 \right\}, \min\left\{ 1 - c_S^{(i)}, F_0^0 \right\} \right\} \right\} \\
s_1^{(i+1)} &= \Bigg| \left| s_0^{(i)} - \min\left\{ \left| s_1^{(i)} - C_1^0 \right|, \left| s_2^{(i)} - C_1^1 \right| \right\} \right| \\
&\qquad - \min\left\{ \left| s_3^{(i)} - k_i \right|, \max\left\{ \min\left\{ c_S^{(i)}, F_1^1 \right\}, \min\left\{ 1 - c_S^{(i)}, F_1^0 \right\} \right\} \right\} \Bigg| \\
s_2^{(i+1)} &= \Bigg| \left| s_1^{(i)} - \min\left\{ \left| s_2^{(i)} - C_2^0 \right|, \left| s_3^{(i)} - C_2^1 \right| \right\} \right| \\
&\qquad - \min\left\{ \left| s_3^{(i)} - k_i \right|, \max\left\{ \min\left\{ c_S^{(i)}, F_2^1 \right\}, \min\left\{ 1 - c_S^{(i)}, F_2^0 \right\} \right\} \right\} \Bigg| \\
s_3^{(i+1)} &= \left| s_2^{(i)} - \min\left\{ \left| s_3^{(i)} - k_i \right|, \max\left\{ \min\left\{ c_S^{(i)}, F_3^1 \right\}, \min\left\{ 1 - c_S^{(i)}, F_3^0 \right\} \right\} \right\} \right|
\end{aligned}
$$

Here, the parts of the register update depending on the control bits ("if c_s then A, else B") have to be expressed by $(c_s \wedge A) \vee (\neg c_s \wedge B)$. For equations not covering the key setup phase, the mixing of s_2 into R and the XOR-ing of the key bits is omitted. Otherwise, the equations are structurally equivalent.

Due to the local updating behaviour of feedback shift registers, the nonsmooth degree of the individual equations ranges between 1 and 5. In particular, this local updating behaviour implies that the degree does not depend on the register length n. By substitution, the number of new equations per clocking can be reduced, thereby trading lower dimensionality for higher degree.

Properties of the Mini-MICKEY equations

By direct calculation, one can verify that the MICKEY equations are not convex. However, as instantiations of the model described by equations (3.11)–(3.16), they are semi-smooth (see also Section 3.3.3). Consequently, it is to be expected that while the optimisation algorithms will quite efficiently converge to stationary points, they need not necessarily be global extrema.

Since most efficient algorithms for estimation of Lipschitz constants only work in the univariate case, we only estimate an upper bound for the Lipschitz constant of the nonsmooth expressions: Each individual expression in the model is Lipschitz-continuous with constant $L = 1$, so for an expression with nonsmooth degree d, the Lipschitz constant is approximated by $L' = 2d$.

3.4.4 Attack scenarios

We considered the following attack scenarios, with the main focus being on the first and third scenario. All scenarios are in a known-plaintext setting, where a sequence of keystream bits can be determined by XOR-ing the known plaintext to the ciphertext.

Scenario 1: Recovering a previous state

In this scenario, given a sequence of keystream bits, the task is to recover the internal state of the stream cipher before the generation of this keystream. This should be infeasible for a secure stream cipher, since knowledge of the full internal state at some point allows the attacker to decrypt all future ciphertexts that are encrypted with keystream derived from this internal state.

From the attacker's point of view, state recovery has the advantage that the loading of the IV and the key can be neglected, which leads to simpler equations. The challenge is that this scenario implies a higher dimension of the equation systems since we need to recover all $2n$ state bits which is typically larger than the length of the key. At least $2n$ bits of keystream are required to carry out this attack.

Scenario 2: Recovering the key

Probably the most obvious target, this scenario encompasses the reconstruction of the k-bit key that was clocked into the cipher during the setup phase from the knowledge of a sequence of keystream bits.

Since for most sensible variants of Mini-MICKEY, $k < 2n$, this has the advantage of a smaller problem dimension, but the disadvantage of higher equation complexity due to the loading of the IV and the key and the preclocking process. At least k bits of keystream are required for this attack.

Scenario 3: Guess and determine the key

This scenario is a special case of the second where a guess for part of the key (say k_0 out of the k bits) is used to simplify the system of equations representing the key loading, preclocking, and the generation of the first $k - k_0$ keystream bits. Since there are only $k - k_0$ unknowns left, the keystream bits are expressed in terms of minimised Boolean functions of the unknown key bits. If this number is sufficiently small, Boolean minimisation algorithms such as the Quine-McCluskey [120] (QMC) or the Espresso method [41] can be applied to simplify the Boolean functions.

In this scenario, very few very complex equations have to be solved. Additionally, the minimisation process is specific to – and therefore has to be repeated for – each key guess. Note also that in this scenario, the overall time complexity can be estimated even if it would be impractical to carry out the attack in practice: If t_0 denotes the time complexity of the minimisation and solving process for one key guess, the total attack complexity is given by $2^{k_0} \cdot t_0$. The attack is faster than a generic attack if t_0 is less than the complexity of $k - k_0$ trial decryptions.

3.4.5 Experimental results

We have subjected the nonsmooth systems of equations corresponding to various versions of Mini-MICKEY and the full MICKEY in the three attack scenarios to different solving algorithms using various approaches to model or adapt the Boolean system of equations. The results are summarised in Tables 3.1 to 3.3 on pages 46–48. The experiments were carried out on a machine with two Intel Xeon dual-core 2.8 GHz processors and 2 GB of memory. Only one core was used, and memory usage was negligible.

Algorithms

The nonsmooth optimisation algorithms used in the experiments were a bundle method (DFBM), the cutting angle method (ECAM), and a dynamical-systems based method (DSO). All algorithms only require the objective function to be Lipschitz, so they are suitable for our nonconvex models of Boolean equations.

DFBM ("Derivative-Free Bundle Method") is a variant of the Bundle algorithm developed by Bagirov and Ugon [8] which uses the method of [7] to approximate the subdifferential. This method yields a particularly efficient procedure for calculating subgradients, so that the running time of the algorithm is commonly dominated by solving the quadratic subproblem (3.19) instead [8]. This algorithm was developed as an extension of [5], especially improving the performance for problem instances of larger dimension. Like all Bundle methods, it converges to local minima. In our experiments, it was combined with random start as a globalisation strategy.

The ECAM ("Extended Cutting Angle Method") intends to improve on DFBM for global optimisation by using the Lipschitz constant of the objective function to estimate its smallest conceivable minimum. Then, a sawtooth-like underestimation of the objective function is constructed [13, 112]. While this method is theoretically guaranteed to find an ε-optimal approximation of the global minimum, it has the disadvantage of typically requiring a large number of function evaluations, and being very sensitive to increases in the dimension of the problem.

The third algorithm, DSO ("Dynamical Systems Based Optimisation"), is a heuristic obtaining descent directions by balancing samples of the objective function according to a physical model of forces [114].

Those algorithms and combinations of them are implemented in the GANSO library for general nonsmooth optimisation developed at the University of Ballarat [6, 14].

Choice of representation

In the case of polynomial models, the choice of representation can have decisive influence on the success of an experiment [127]. By contrast, this has little effect in the nonsmooth case, where the different nonsmooth representations corresponding to standard and Fourier representation and their respective duals all caused very comparable behaviour of the algorithms. Specifically, they all exhibit the property that there is no one-to-one correspondence of solutions to $a \oplus b = 0$ between Boolean values and the unit interval, frequently leading the solvers to suggest non-Boolean values as "improvements" over non-matching Boolean points of iteration.

We therefore focus on the standard representation given by equations (3.11)–(3.16).

Choice of norm

While the choice of representation had little effect on the behaviour of the minimisation algorithms, the choice of norm has a much greater impact. For all of the algorithms in the GANSO library, the maximum norm turned out to be the best choice.

Equation preprocessing

The success of the solvers can depend significantly on how an optimisation problem is formulated. We investigated the effect of some equation preprocessing techniques to adapt the system of equations to the solver.

Guessing different subsets of key bits. If the attack scenario involves the guessing of key bits, the key material can be selected in an anticipatory way to maximise the simplifying effect on the system of equations. It turns out that guessing key bits between $k_{\lfloor n/3 \rfloor}$ and $k_{\lfloor 2n/3 \rfloor}$ has the biggest impact regarding simplification, since on the one hand, the first key bits are clocked in independently of the state, and on the other hand, the "avalanching" effect of the simplifications is weaker for the last part of the key.

Trading equation complexity for dimensionality. The straightforward method of generating equations representing the stream cipher clocking results in far more individual equations than variables. In case of the nonsmooth optimisation algorithms we applied, it usually turned out to be preferable combining many equations by substitution to reduce the dimensionality of the problem. Since only the expression nesting level (but not the degree) is affected, the resulting increase in equation complexity is usually tolerable.

Experimentally, the optimal balance between dimensionality and equation complexity was reached by organizing the individual subexpressions according to a balanced tree structure such that a maximum substitution level of about five subequations is not exceeded.

Ensuring Booleanness of the solutions without affecting convergence. To mitigate the issue with non-Boolean solutions suggested by the solvers, two approaches were considered:

The first involves adding a penalisation $P_\alpha(x)$ to the objective function, which then becomes $\|f(x)\| + P_\alpha(x)$, introducing a penalty increasing with the distance to the corners of the hypercube $[0,1]^n$ specified by the box constraints. In particular, the following terms were considered, with α being a parameter to fine-tune the weight of the penalty term:

$$\text{a triangle-like function:} \qquad x \mapsto \alpha \cdot \left({}^1\!/\!{}_2 - |x - {}^1\!/\!{}_2|\right), \tag{3.23}$$

$$\text{a quadratic penalty term:} \qquad x \mapsto \alpha \cdot \left(x - x^2\right), \tag{3.24}$$

$$\text{an exponential bell curve:} \qquad x \mapsto \alpha \cdot \left(e^{-8\cdot(2x-1)^2}\right). \tag{3.25}$$

As with the choice of norms, the maximum, sum, or average of all component deviations from $\{0,1\}$ can be used in $P_\alpha(x)$. These penalty terms are illustrated in Figure 3.6.

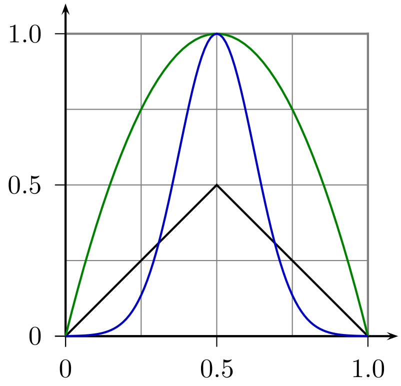


Figure 3.6: Penalty terms to reward Booleanness.

All of the above can also be combined with multiplication by x or square rooted, the latter resulting in a steeper penalisation. In most cases, a damped exponential or a plain triangle penalizer were yielding the best results. This approach has the obvious disavantage of causing convergence issues if the penalty terms modify the notion of descent too much. Its main merit is that it does not increase the dimension of the problem.

The second approach is to introduce a second set of equations of the form $x^2 = x$, directly forcing the variables to be Boolean. This has the advantage of not modifying the objective function, so that convergence is not affected by the penalisation terms, but increases the dimension. Furthermore, those equations introduce additional local optima which might hinder convergence to a global optimum. For small instances with a dimension of less than 20, this approach turned out to be superior to the penalty terms, which in turn start to yield better results for greater dimensions.

3.4.6 The DDT heuristic and MICKEY

In addition to the nonsmooth algorithms, we applied the DDT heuristic to the systems of equations of Mini-MICKEY. The implementation used was a simplified variant of DDT as implemented by [78]. The experimental data from Table 3.4 suggests that this approach is inferior to the nonsmooth algorithms. However, extensions to this implementation of DDT are expected to improve the results.

3.4.7 Discussion

Results

In the state recovery scenario, due to the comparatively high dimension of this problem, only very small instances ($n = 6$, resulting in a 72-dimensional instance) could be fully solved. For higher values of n, all solvers failed. In most of those cases, non-Boolean coordinates were output which could not be mapped back to a Boolean solution by rounding or similar procedures. A closer inspection of the final points of iteration revealed that all of the solvers got stuck in local optima. It is worth noting that longer keys do not have any significant influence in this scenario since we attempt to recover the state after the key has been clocked in.

Concerning key recovery, the results were more promising. The maximum problem size that the nonsmooth algorithms could fully solve were $n = 20$ and a 26-bit key. In those cases, additional equations of the form $x^2 = x$ were required to ensure Booleanness, resulting in a total dimension of 52.

In both scenarios, the results are given for the maximum key length for a particular state size for which the problems could still be solved. In particular, this means that all instances involving the same state but smaller key sizes can be successfully solved.

n	k	Solver	Preprocessing	Execution time	Function eval.	Solvable
4	4	DFBM	–	$\ll 1$ s	2^{9}	no[a]
4	4		triangle penalty	$\ll 1$ s	2^{10}	yes
4	8		triangle penalty	$\ll 1$ s	$2^{11.5}$	yes
6	6		triangle penalty	15 s	2^{13}	yes
6	16		triangle penalty	15 s	$2^{13.2}$	yes
7	7		triangle penalty	28 s	2^{16}	no[b]
8	8		triangle penalty	149 s	$2^{17.5}$	no[b]
9	9		triangle penalty	3 min	2^{19}	no[c]
9	9		exp. penalty	3 min	2^{19}	
9	9		triangle penalty + balancing	3 min	2^{19}	
4	4	ECAM	–	2.2 s	$2^{10.5}$	no[d]
4	4		triangle penalty	2.3 s	$2^{10.7}$	yes
4	8		triangle penalty	2.3 s	2^{11}	yes
6	6		triangle penalty	52 s	$2^{14.4}$	yes
6	16		triangle penalty	52 s	2^{15}	yes
7	7		triangle penalty	7 min	$2^{17.6}$	no[a]
7	7		triangle penalty + balancing	12 min	$2^{19.5}$	no[a]
4	4	DSO	–	$\ll 1$ s	2^{8}	yes
4	8		–	$\ll 1$ s	2^{9}	yes
6	6		–	$\ll 1$ s	$2^{12.1}$	yes
6	16		–	$\ll 1$ s	$2^{12.7}$	yes
7	7		–	17 s	$2^{15.4}$	no[a]
7	7		triangle penalty	17 s	$2^{16.2}$	no[a]
7	7		triangle penalty + balancing	17 s	2^{17}	no[a]

[a] Solutions not Boolean.
[b] Solvable for specific initial values only.
[c] No improvement if more iterations are used.
[d] Convergence is unreliable.

Table 3.1: State recovery for n bit registers and k bit key. Dimension is $2n$.

In the case of combining key recovery with Boolean minimisation according to a partial key guess, both the instances involving a key guess of 72 and 64 bits for the full MICKEY could be solved. The DSO solver proved particularly suitable for these problem instances. The increase in nonsmooth degree per equation

n	k	Solver	Preprocessing	Execution time	Function eval.	Solvable
4	4	DFBM	triangle penalty	$\ll 1$ s	2^{5}	yes
4	8		triangle penalty	$\ll 1$ s	$2^{8.2}$	yes
8	8		triangle penalty	6 s	$2^{8.4}$	yes
12	12		triangle penalty	9 min	$2^{13.1}$	yes
12	20		triangle penalty	16 min	$2^{21.3}$	yes
16	20		triangle penalty	20 min	$2^{21.6}$	yes
20	20		exp. penalty	20 min	$2^{22.1}$	yes
20	26		exp. penalty + $x^2 = x$	43 min	$2^{28.4}$	yes
20	27		exp. penalty + $x^2 = x$	43 min	2^{29}	no[b]
4	4	ECAM	exp. penalty	2 s	$2^{5.7}$	yes
8	8		exp. penalty	15 s	$2^{10.8}$	yes
16	20		exp. penalty + balancing	94 min	$2^{23.1}$	yes
20	22		exp. penalty + balancing	111 min	$2^{25.2}$	no[c]
4	4	DSO	–	$\ll 1$ s	$2^{4.1}$	yes
8	8		–	$\ll 1$ s	$2^{8.4}$	yes
16	20		–	14 min	$2^{21.9}$	no[a]

[a] Solutions not Boolean.

[b] Solvable for specific initial values only.

[c] Convergence is unreliable.

Table 3.2: Key recovery for n bit registers and k bit key. Dimension is k.

turned out to be comparatively small: 8 for the 72-bit guess and 11 for the 64-bit guess. This is a result of the good Boolean minimisation performed by the Quine-McCluskey and Espresso algorithms. Additionally, the success rate was improved at the cost of running time by preprocessing the equations such that the maximum degree of 5 per equation was not exceeded.

It should be noted that the success of the solver in those cases does not yield a valid attack on the full MICKEY. The whole process of generating and minimising the equations has to be repeated for each key guess, which is clearly inferior to brute force. It is an open issue to extend this to more useful values of the number of guessed key bits.

k_0	Solver	Preprocessing	Execution time	Function eval.	Solvable
76	DFBM	exp. penalty	26 s	$2^{5.4}$	yes
72		exp. penalty	13 min	$2^{9.8}$	yes
68		exp. penalty	24 min	$2^{13.1}$	yes
64		exp. penalty	59 min	2^{18}	no[a]
76	ECAM	exp. penalty	33 s	$2^{5.2}$	yes
72		exp. penalty	1 min	$2^{9.5}$	yes
68		exp. penalty	7 min	$2^{12.8}$	no[a]
76	DSO	–	17 s	2^{5}	yes
72		–	54 s	$2^{9.2}$	yes
68		–	2 min	$2^{12.2}$	yes
64		–	4 min	$2^{17.3}$	yes
60		–	11 min	$2^{22.5}$	no[a]
60		triangle penalty	11 min	2^{23}	no[a]
60		exp. penalty	11 min	$2^{23.1}$	no[a]
60		exp. penalty + $x^2 = x$	11 min	$2^{25.5}$	no[a]

[a] Solutions not Boolean.

Table 3.3: Guess k_0 out of 80 key bits and determine the remaining key bits for the full MICKEY ($n = 100$ bit registers and $k = 80$ bit key). Dimension is $80 - k_0$. Balancing was used in all experiments.

Notes on the nonsmooth optimisation algorithms

The experimental results provided several insights about the applicability of the different nonsmooth optimisation algorithms to this type of systems of equations.

Generally, it could be observed that the dynamic systems based method (DSO) was particularly efficient for instances of small dimension. This algorithm was far less sensitive to the complexity of the individual expressions, while increasing the dimension proved problematic. By contrast, the DFBM and ECAM algorithms could still successfully solve some instances with higher dimension. This is due to the fact that the equation systems in the state and (full) key recovery scenarios had higher dimension but relatively small nonsmooth degree, which particularly also bounded the Lipschitz constant to

		State recovery		
n	k	Preprocessing	Execution time	Solvable
4	4	triangle penalty	$\ll 1$ s	yes
		Key recovery		
n	k	Preprocessing	Execution time	Solvable
4	4	triangle penalty	$\ll 1$ s	yes
4	8	triangle penalty	$\ll 1$ s	yes
		Key guess and determine $(n = 100, k = 80)$		
	k_0	Preprocessing	Execution time	Solvable
	76	exp. penalty	26 s	yes
	72	exp. penalty	3 min	yes
	68	exp. penalty	4 min	no (local optimum)
	64	exp. penalty	9 min	no (local optimum)

Table 3.4: Experimental results applying the DDT heuristic to the three attack scenarios.

a small value.

The DDT heuristic for pseudo-Boolean programming turned out to be an algorithm with reliable running time polynomial in the problem dimension, however at the expense of guarantees about the quality of the solution. The Bundle and ECAM methods provided rather unreliable running times, but the quality of the solution improved with the number of iterations. Especially the ECAM method, however, can take a prohibitive number of steps before reaching an ε-optimal solution, which in the context of cryptanalysis is only then an advantage if the ε-optimal solution can be mapped back to the sought Boolean solution.

3.5 Conclusions

Summarising, the concept of nonsmooth models of Boolean equations is a generic approach that can be used in the cryptanalysis of any algorithm based on arithmetic over $\mathbb{F}_2$, including most stream ciphers, block ciphers and hash functions. The approach aims at avoiding real-valued expressions of higher degree by using nondifferentiable but Lipschitz-continuous functions. The theory of and available numerical software for nonsmooth optimisation can then be used to attempt solving the corresponding systems of equations.

Applied to the stream cipher MICKEY from the eSTREAM final portfolio, this approach can recover the 80-bit key of the full algorithm in 2^{64} iterations if 16 bits of keystream are available. However, while this attack requires very little known plaintext, the total time complexity is slightly inferior to trying all 2^{80} keys. Small-scale variants of MICKEY can be solved in practical time without guessing key bits.

With the exception of the guess-and-determine scenario, one main issue with this approach compared to statistical attacks is the lack of predictability beyond practical attacks: An attack can only be shown to work if it can actually be carried out in practice. Extrapolating results from small-scale experiments to the full algorithms is an unreliable measure since the number of required iterations for a global optimum could grow exponentially (ECAM), or the algorithms could converge to local extrema without ever reaching the solution corrresponding to the key or the previous state (DSO, DFBM).

However, these techniques could be applied to the various equation-solving subproblems occurring in typical cryptanalytic attacks where ad hoc or guess-and-determine techniques are currently used, for instance within collision attacks on SHA-1 or SHA-2 [93]. Those subproblems typically have lower dimensionality than full ciphers or hash functions, and improving the time required to solve them would directly improve the full attack complexities. For instance, [2] demonstrates several methods of how equation-solving techniques over finite fields (using Gröbner bases and SAT solvers) can be used to enhance differential attacks on block ciphers.

Chapter 4

Hash functions and the rebound attack

This chapter is dedicated to the interplay between the design of hash functions based on the wide trail design strategy [56] and the rebound attack [122], a recent attack strategy specifically developed for such constructions.

4.1 Wildpool: Difference propagation in unaligned diffusion layers

The most well-known hash function following the wide trail strategy is Whirlpool by Barreto and Rijmen [12]. It features a compression function built from a 512-bit block cipher W using the Miyaguchi-Preneel construction. W itself is a 10-round iterated block cipher with a round function designed according to the wide trail strategy, allowing bounds for its resistance against differential and linear cryptanalysis to be proven.

In a design study by Nikova [129], a variant of Whirlpool was constructed based on the idea of employing a larger S-box of 16 bits, versus 8 bits in Whirlpool. Since in a typical wide trail design, the linear diffusion layer is an MDS matrix [144] implemented using finite field arithmetic, it was considered desirable to keep the diffusion layer operating on quantities of 8 bits for efficiency. This leads to an unusual and new situation in wide trail designs:

The confusion and diffusion layers are no longer operating on quantities of the same size, they become “unaligned”. The consequences of this are studied in this section.

4.1.1 Brief description of Wildpool

We give a brief description of the Wildpool specification [129], focusing on the aspects relevant for our analysis. Most of its components are identical to Whirlpool.

Wildpool produces a 512-bit hash value. Its compression function is based on a dedicated block cipher VV, which operates on a 512-bit *hash state* using a 512-bit *chained key state*, both of which are internally viewed as an 8×8 matrix over $\mathbb{F}_2^8$, indexed as $a_{i,j}$ for $0 \leq i, j < 8$. Analogous to Whirlpool the Miyaguchi-Preneel construction is used.

The round transformation of VV is defined by operating on a state matrix M using the key matrix k as follows:

$$\rho[k] = \sigma[k] \circ \theta \circ \pi \circ \gamma.$$

$\sigma[k]$ denotes the addition of the round keys via bitwise XOR. γ is a non-linear substitution box $S : \mathbb{F}_2^{16} \to \mathbb{F}_2^{16}$, defined by the inverse $x \mapsto x^{-1}$ in $\mathbb{F}_2^{16}$, operating on the two bytes i and $i+4$ for $i = 1, \ldots, 4$ in each row of the state matrix, taken as a 16-bit quantity. An important property of S is that it is defined using a special normal basis representation of $\mathbb{F}_2^{16}$ over $\mathbb{F}_2^8$. For details on the normal basis representation, we refer to Section 4.5.

π is a cyclic permutation identical to the one in Whirlpool, cyclically shifting each column of the state matrix independently, such that column j is cyclically shifted downwards by j positions.

Like π, the linear diffusion layer θ is identical to Whirlpool and consists of the application of an 8×8 circulant MDS matrix [144] to each row of the state. Its branch number is $\mathcal{B}(\theta) = 9$. Together, π and θ form the diffusion layer of Wildpool.

The internal block cipher VV consists of 10 repetitions of the round transformation.

4.1.2 Difference cancellation in unaligned diffusion layers

The fact that both π and θ operate on halves of each S-box output implies that from the perspective of the diffusion layer, every application of an S-box is

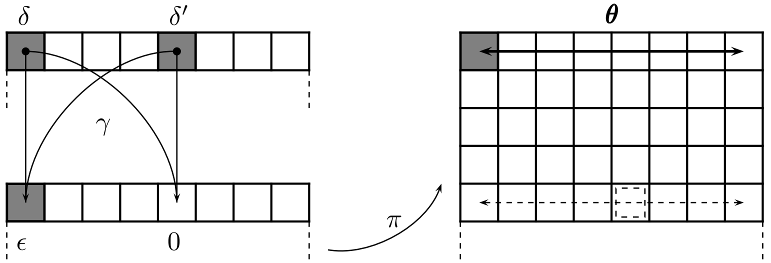


Figure 4.1: Half-zero output difference causing deactivation of rows.

already combining two adjacent 8-bit cells, which can potentially have negative impact on overall diffusion.

A first observation in that direction is that it is now conceivable to have a nonzero difference in two 8-bit input cells to one S-box that turns into a zero difference in one of the two output cells, consequently possibly deactivating one row after the application of π, as illustrated in Figure 4.1. Note that such a case is impossible in a standard wide trail design (such as Whirlpool) where diffusion and confusion layer are operating on the same granularity, since a bijective S-box has zero output difference precisely when the input difference was already zero.

This observation motivates the definition of *half-zero* differences.

Definition 4.1. Let $S : (a, b) \mapsto (c, d)$ denote the inversion mapping in $\mathrm{GF}(2^{2m})$. We say that a pair $(a_1, b_1) \neq (a_2, b_2)$ with a nonzero input difference causes a *half-zero output difference* if either

$$S(a_1, b_1) + S(a_2, b_2) = (a', 0) \quad \text{for some } a' \neq 0 \tag{4.1}$$

or

$$S(a_1, b_1) + S(a_2, b_2) = (0, b') \quad \text{for some } b' \neq 0 \ . \tag{4.2}$$

The number of distinct pairs fulfilling (4.1) resp. (4.2) are denoted $N_r(m)$ resp. $N_l(m)$.

It turns out that such patterns (with δ and δ' both nonzero) exist in Wildpool's S-box. Additionally, there are patterns where a half-zero input difference maps to some half-zero output difference, and for any given input difference (δ, δ'), there is at least one possible output difference with a zero upper and one with a zero lower half.

In general, the occurrence of half-zero difference propagations is an inherent property of the inversion mapping in any finite field $\mathbb{F}_2^{2m}$. To analyse this, we make use of the following formula for the inverse of $\mathbb{F}_2^{2m}$ expressed over $\mathbb{F}_2^m$ (see Section 4.5.3):

$$(c,d) = (a,b)^{-1} \Leftrightarrow \begin{cases} c = \left((a+b)^2 gq + q^2 ab\right)^{-1} \cdot b \\ d = \left((a+b)^2 gq + q^2 ab\right)^{-1} \cdot a \end{cases} , \tag{4.3}$$

with q and g denoting two constant elements of $\mathbb{F}_2^m$ depending on the basis choice (see Section 4.5.2). Here, we will only use the property that both q and g are fixed nonzero elements of $\mathbb{F}_2^m$.

The number of cases where the difference in the right half attains zero given a nonzero input difference can therefore be described as the number of pairs (a_1, b_1) and (a_2, b_2) with $(a_1, b_1) \neq (a_2, b_2)$ such that

$$\left((a_1+b_1)^2 gq + q^2 a_1 b_1\right)^{-1} b_1 = \left((a_2+b_2)^2 gq + q^2 a_2 b_2\right)^{-1} b_2,$$

which is equivalent to

$$\left((a_2+b_2)^2 gq + q^2 a_2 b_2\right) b_1 = \left((a_1+b_1)^2 gq + q^2 a_1 b_1\right) b_2 \ . \tag{4.4}$$

Now fix a pair (a_1, b_1). To see how many pairs (a_2, b_2) satisfying equation (4.4) exist, we set $a := a_2, b := b_2, c := gq(a_1+b_1)^2 + q^2 a_1 b_1$ and $d := b_1$ which yields

$$gqda^2 + q^2 dba + gqdb^2 + bc = 0. \tag{4.5}$$

For fixed b, this is a quadratic equation in a. Rewriting this in standard form:

$$a^2 + a + \underbrace{g(gbd + c/q)}_{=:k} = 0, \tag{4.6}$$

we see that (4.6) has two solutions a_0, a_1 if and only if $\mathrm{Tr}(k) = 0$ and no solutions otherwise [111].

To count the number of times the expression

$$\mathrm{Tr}(k) = \mathrm{Tr}\left(g^2 b_1 b_2 + g^2 (a_1+b_1)^2 + gqa_1 b_1\right) \tag{4.7}$$

evaluates to zero when a_1, b_1, b_2 range over $\mathrm{GF}(2^m)$, we use the following properties of the trace:

Lemma 4.2. *Let $m, n > 0$, $q = p^n$ any prime power and denote $F = GF(q^m)$, $K = GF(q)$. Then the following holds.*

(i) In any basis $\{b_0, \ldots, b_{m-1}\}$ of F over K, there exists at least one i, $0 \leq i < m$, such that $Tr_{F/K}(b_i) \neq 0$.

(ii) As x ranges over $K = GF(p^n)$, $Tr_{GF(p^n)/GF(p)}(x)$ takes on each element of $GF(p)$ equally often, namely p^{n-1} times.

Proof. (i) We first show that F must contain an element a_1 such that $\mathrm{Tr}_{F/K}(a_1) \neq 0$. To see this, note that $\mathrm{Tr}_{F/K}(x) = 0$ if and only if x is a root of the polynomial $\sum_{i=0}^{m-1} x^{q^j}$ which can have no more than q^{m-1} roots, whereas F contains q^m elements.

To prove the claim, suppose that $\mathcal{B} := \{b_0, \ldots, b_{m-1}\}$ is a basis of F over K such that $\mathrm{Tr}_{F/K}(b_i) = 0$ for all i. Since $\mathcal{B}$ is a basis, any element a of F can be written as $a = \sum_{i=0}^{m-1} a_i b_i$ with coefficients $a_i \in K$. This however implies

$$\begin{aligned}
\mathrm{Tr}_{F/K}(a) &= \mathrm{Tr}_{F/K}\left(\sum_{i=0}^{m-1} a_i b_i\right) \\
&= \mathrm{Tr}_{F/K}(a_0 b_0) + \cdots + \mathrm{Tr}_{F/K}(a_{m-1} b_{m-1}) \\
&= a_0 \mathrm{Tr}_{F/K}(b_0) + \cdots + a_{m-1} \mathrm{Tr}_{F/K}(b_{m-1}) \\
&= 0,
\end{aligned}$$

so that all linear combinations of the b_i have a zero trace. Since F contains at least one element with a nonzero trace, the b_i do not span F and hence cannot be a basis. Consequently, at least one of the basis elements must have a nonzero trace.

(ii) Choose any basis $\{b_0, \ldots, b_{n-1}\}$ of $\mathrm{GF}(p^n)$ over $\mathrm{GF}(p)$ and write each $x \in \mathrm{GF}(p^n)$ as $x = \sum_{i=0}^{n-1} x_i b_i$. As above, we have

$$\mathrm{Tr}(x) = x_0 \mathrm{Tr}(b_0) + \cdots + x_{n-1} \mathrm{Tr}(b_{m-1}) \, .$$

Define $I_0 := \{i \mid \mathrm{Tr}(b_i) = 0\}$ and $I_1 := \{i \mid \mathrm{Tr}(b_i) \neq 0\}$. Because of property (i), at least one of the b_i must have a nonzero trace, so that I_1 is nonempty. The trace of x is then

$$\mathrm{Tr}(x) = \sum_{i \in I_1} x_i \mathrm{Tr}(b_i) \, , \tag{4.8}$$

and as multiplication with a nonzero constant is a bijection, each of the summands takes on all values of $\mathrm{GF}(p)$ exactly once when x_i ranges over $\mathrm{GF}(p)$. As $(\mathrm{GF}(p), +)$ is a finite group, its Cayley table is a latin square and hence the sum (4.8) takes on all values of $\mathrm{GF}(p)$ exactly $p^{|I_1|-1}$ times when the x_i $(i \in I_1)$ range over $\mathrm{GF}(p)$. As x ranges over $\mathrm{GF}(p^n)$, this is repeated $p^{|I_0|}$ times, and since $n = |I_0| + |I_1|$, the result follows immediately. ☐

For any fixed pair (a_1, b_1), the expression

$$g^2 b_1 b_2 + g^2(a_1 + b_1)^2 + gqa_1 b_1 \tag{4.9}$$

is an affine mapping in b_2, i.e. equal to $g^2 b_1 \cdot b_2 + C$ for a constant C. This mapping is bijective if and only if $b_1 \neq 0$. In this case, according to Lemma 4.2, there are 2^{m-1} values of b_2 for each fixed choice of $(a_1, b_1), b_1 \neq 0$ where (4.7) is zero, accounting for $2^m \cdot (2^m - 1) \cdot 2 \cdot 2^{m-1} = 2^{3m} - 2^{2m}$ pairs causing a half-zero output difference.

If, on the other hand, $b_1 = 0$, expression (4.9) reduces to $g^2 a_1^2$, and we have $\operatorname{Tr}\bigl(g^2 a_1^2\bigr) = \operatorname{Tr}\bigl((ga_1)^2\bigr) = \operatorname{Tr}(ga_1)$. Since $g \neq 0$ by construction, the mapping $a_1 \mapsto ga_1$ is a permutation of $\mathrm{GF}(2^m)$. By Lemma 4.2, $\operatorname{Tr}(ga_1) = 0$ for 2^{m-1} values of a_1, and since this is independent of the value of b_2, we have another $1 \cdot 2^m \cdot 2 \cdot 2^{m-1} = 2^{2m}$ pairs for the case $b_1 = 0$.

Altogether, there are $2^{3m} - 2^{2m} + 2^{2m} = 2^{3m}$ pairs fulfilling condition (4.4), whereof 2^{2m} pairs with a zero input difference (i.e. $(a_1, b_1) = (a_2, b_2)$) have to be subtracted for our purposes. Consequently, the number of pairs with nonzero input difference causing a zero output difference in the second output half is $2^{3m} - 2^{2m}$. The case of a zero output difference in the first half is completely analogous. This proves

Theorem 4.3. *For the inversion mapping in any finite field $GF(2^{2m})$, the number of distinct pairs $(a_1, b_1) \neq (a_2, b_2)$ causing a half-zero output difference verifies*

$$N_r(m) = N_l(m) = 2^{3m} - 2^{2m} \quad .$$

4.1.3 Proving bounds

The main consequence of observation of the previous section is that the security analysis of Whirlpool cannot directly be applied to Wildpool due to the existence of half-zero differences. In order to obtain a lower bound on the number of active S-boxes, we reorder the columns such that the two cells corresponding to one S-box become adjacent and consider the whole state as a 8×4 matrix A': For $0 \leq i < 8$ and $0 \leq j < 4$, we have $a'_{i,j} = \begin{pmatrix} a_{i,j} & a_{i,j+4} \end{pmatrix}$, see also Figure 4.2. Note that this just constitutes a modified view; the operations remain the same.

In this modified view, it is clear that a active S-boxes in one row can correspond to $a, \dots, 2a$ active 8-bit cells, and c active cells imply at least $\lceil c/2 \rceil$ active S-boxes in that row. The branch number $\mathcal{B}'(\theta)$ with regard to the 8×4 structure can now be determined as follows: Consider a (row) vector $v \in (\mathbb{F}_2^{16})^4$. Suppose its Hamming weight to be $w_h(v) = a > 0$, so that there are a active S-boxes in

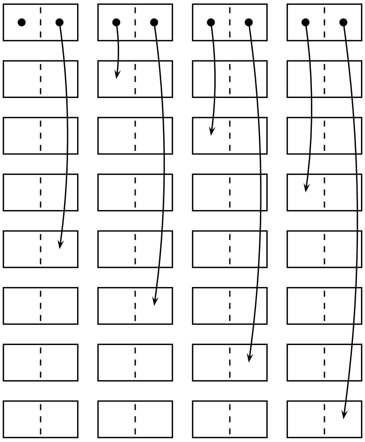


Figure 4.2: Modified view on Wildpool's state. The arrows indicate the operation of π.

this row, corresponding to a number of c active cells, where $a \leq c \leq 2a$. Since $\mathcal{B}(\theta) = 9$, there are at least $9 - c$ active cells and hence at least $\lceil (9-c)/2 \rceil$ active S-boxes in the image of v. Consequently,

$$w_h(v) + w_h(\theta(v)) \geq a + \lceil (9 - (2a))/2 \rceil = 5$$

for any nonzero v. Since 5 is also an upper bound, we have $\mathcal{B}'(\theta) = 5$.

Since π distributes the output of different S-boxes to different rows also in the modified setting, it fulfills the criterion of *diffusion optimality* formulated in [55]. Hence, Theorem 2 of [55] is applicable, implying a lower bound of $\mathcal{B}'(\theta)^2 = 25$ for the number of active S-boxes in four rounds. Together with the fact that the maximum differential probability of a 16-bit inversion S-box is 2^{-14} [132], any differential trail over four-rounds has therefore a maximum probability of $(2^{-14})^{25} = 2^{-350}$. This has to be compared to the four-round bound of $2^{-5 \cdot 81} = 2^{-405}$ in the case of Whirlpool: despite the bigger S-box, the security bound that can be proven is lower than for Whirlpool. This is due to the effect of the unaligned diffusion layer.

4.2 The rebound attack

In this section, we give a brief introduction to the rebound attack. It is an advanced differential attack tailored towards the cryptanalysis of hash functions, where there is no secret key, and the attacker can start computing anywhere in an iterated mapping. It has been published by Mendel*et al.* [122] with applications to Whirlpool and Grøstl [75]. Since its invention, it has been used in the cryptanalysis of many hash functions. Being especially suitable for the analysis of wide trail designs, it has among others been applied to Maelstrom [122], ECHO [94, 121, 150] and LANE [119]. But also designs based on modular addition, rotation and XOR have been cryptanalysed using Rebound techniques, for instance the hash function Skein [100].

In its basic form, the idea of a rebound attack is to split the algorithm F under consideration (a compression function, an internal block cipher or an internal permutation of a hash function) into three parts:

$$F = F_{fwd} \circ F_{in} \circ F_{bwd}.$$

The middle part is called *inbound phase*, the inner and outer part are called *outbound phases*. This is due to the fact that the attacker starts at the inbound phase, calculating "inwards" from both ends of the inbound phase to generate solutions for a desired differential transition over the inbound part. Those solutions are then probabilistically propagated forward through F_{fwd} and backwards through F_{bwd} according to certain truncated differential paths. Its main advantage is that by placing the most expensive part of the differential trail in the middle (the inbound phase), the freedom stemming from the values of the internal state variables can be efficiently used to obtain solutions to this part with high probability. For a detailed description, we refer to the original paper [122]. Moreover, variants of the basic rebound attack will be discussed in Sections 4.4.3 and 4.6.3.

Note also that there exist advanced variants [119, 150] that consist of multiple inbound and multiple outbound phases.

The rebound attack has been successfully applied to Whirlpool [106, 122]: For the hash function, near-collisions can be found for 7.5 of the 10 rounds using 2^{128} time and 2^{64} memory. For all 10 rounds of the compression function, there exists a distinguisher with complexity 2^{188} time and 2^{8} time. Due to the similarity between Whirlpool and Wildpool, all these attacks also apply to Wildpool.

4.3 Whirlwind: Designing a rebound-resistant hash function

Since both Whirlpool and the initial design study Wildpool can be successfully attacked by rebound techniques, it remains an interesting problem to construct a wide trail hash function that does not succumb to the rebound attack.

It turns out that a decisive contributing factor to the success of rebound attacks on Whirlpool (and thus Wildpool) is the presence of the key schedule [150]. It provides a crucial opportunity to influence the state variables since it introduces additional degrees of freedom available to the attacker.

Intuitively, long sequences of round transformations without influence on intermediate values via the XOR of a key, a message or a feedforward makes rebound attacks harder to apply. In a wide trail design, the inbound phase is limited to 2-3 rounds [77, 150], and the probability of the truncated differential trails for the outbound phases quickly diminishes with an increased number of rounds, since according to the optimal diffusion property of the wide trail strategy, an additional state with all S-boxes active cannot be avoided for an extended number of rounds.

This observation leads us to the design of the Whirlwind hash function [11]: Whirlwind is a hash function that (somewhat loosely) follows the Sponge model [16]. Its compression function is based on the repeated application of a round transformation, similar to a block cipher, and designed according to the wide trail strategy [56, 57], and inspired by Whirlpool. Like Wildpool, an unaligned diffusion layer is being used, however, in a refined way that allows better bounds on trails to be proven.

4.3.1 Description of Whirlwind

In this section, we give an overview of the Whirlwind specification [11].

Internal state and round transformation

Whirlwind has an internal state of 1024 bits, which can be represented by an 8×8 array of 16-bit elements:

$$a = [a_{i,j}]_{i,j=0}^{7} .$$

Each of the 16-bit elements of the state can in turn be represented by a 2×2 array of 4-bit elements:

$$\forall i, j : \; a_{i,j} = \begin{bmatrix} a_{i,j,0,0} & a_{i,j,0,1} \\ a_{i,j,1,0} & a_{i,j,1,1} \end{bmatrix}$$

The i-th row of the state is denoted by a_i or equivalently by $a_{i,*,*,*}$. These 4-bit elements are the smallest entity used in Whirlwind.

The 4-bit elements in Whirlwind are expressed as elements of $\mathrm{GF}(2^4)$ in terms of a tower field decomposition to $\mathrm{GF}(2^2)$ and $\mathrm{GF}(2^2)$ using normal bases at each level. For the exact specification of this decomposition, we refer to the paper [11].

The round transformation comprises four maps, which we describe in turn below.

The nonlinear substitution layer γ. The nonlinear layer operates in parallel on the 64 elements of 16 bits by replacing each 16-bit element by its multiplicative inverse:

$$\gamma : \mathrm{GF}(2^{16})^{8\times 8} \to \mathrm{GF}(2^{16})^{8\times 8} : [a_{i,j}]_{i,j=0}^{7} \mapsto \left[(a_{i,j})^{-1}\right]_{i,j=0}^{7}.$$

We additionally define the inverse of zero to be zero.

The linear maps θ and τ. The first linear map τ is defined as a transposition of the 8×8 state matrix:

$$\tau : \mathrm{GF}(2^{16})^{8\times 8} \to \mathrm{GF}(2^{16})^{8\times 8} : [a_{i,j}]_{i,j=0}^{7} \mapsto [a_{j,i}]_{i,j=0}^{7}$$

Note that the explicit implementation of τ can be avoided by implementing two different round transformations: one where θ acts on the rows, and one where θ acts on the columns.

The second linear map θ operates on the 8 rows of the state in parallel by applying a linear function λ to each row.

$$\theta : \mathrm{GF}(2^{16})^{8\times 8} \to \mathrm{GF}(2^{16})^{8\times 8} : [a_{i,j}]_{i,j=0}^{7} \mapsto [\lambda(a_i)|_j]_{i,j=0}^{7}.$$

Furthermore, λ acts in parallel on the 4-bit subcomponents of the $a_{i,j}$:

$$\lambda(a_i) = b_i \Leftrightarrow \begin{cases} \lambda_0(a_{i,*,0,0}) = b_{i,*,0,0} \\ \lambda_1(a_{i,*,0,1}) = b_{i,*,0,1} \\ \lambda_1(a_{i,*,1,0}) = b_{i,*,1,0} \\ \lambda_0(a_{i,*,1,1}) = b_{i,*,1,1} \end{cases}$$

The maps λ_0, λ_1 are defined as follows:

$$\lambda_0 : \quad \mathrm{GF}(2^4)^{1\times 8} \to \mathrm{GF}(2^4)^{1\times 8} : a_{i,*,k,k} \mapsto a_{i,*,k,k} \cdot M_0$$

$$\lambda_1 : \quad \mathrm{GF}(2^4)^{1\times 8} \to \mathrm{GF}(2^4)^{1\times 8} : a_{i,*,k,1-k} \mapsto a_{i,*,k,k} \cdot M_1$$

with

$$M_0 = \text{dyadic}\left(\mathtt{5}_x, \mathtt{4}_x, \mathtt{A}_x, \mathtt{6}_x, \mathtt{2}_x, \mathtt{D}_x, \mathtt{8}_x, \mathtt{3}_x\right) \qquad \text{and}$$

$$M_1 = \text{dyadic}\left(\mathtt{5}_x, \mathtt{E}_x, \mathtt{4}_x, \mathtt{7}_x, \mathtt{1}_x, \mathtt{3}_x, \mathtt{F}_x, \mathtt{8}_x\right),$$

where dyadic (s) denotes the dyadic matrix S corresponding to the sequence s over $\mathrm{GF}(2^4)$, i.e. $S_{i,j} = s_{i \oplus j}$.

This form of an unaligned diffusion layer ensures that the functions λ, θ inherit the optimal diffusion properties of the λ_i maps. Note however, that when they are described as acting on elements of $\mathrm{GF}(2^{16})$, like the other components of the round transformation, then this requires the use of a linearised polynomial [111], instead of a simple matrix multiplication. This has the effect of making the algebraic description of the round transformation over $\mathrm{GF}(2^{16})$ more complex.

The affine layer σ^r. In this layer, a round-dependent constant c^r is added to the state, in order to break the symmetry between different positions in the state.

$$\sigma^r : \mathrm{GF}(2^{16})^{8\times 8} \to \mathrm{GF}(2^{16})^{8\times 8} : [a_{i,j}]_{i,j=0}^{7} \mapsto \left[a_{i,j} + c^r_{i,j}\right]_{i,j=0}^{7},$$

with $c^r = \gamma(s^r)$ and

$$\begin{array}{rcll} s^r_{0,j} & = & 8(r-1)+j, & 0 \le j \le 7, \\ s^r_{i,j} & = & 0, & 1 \le i \le 7, 0 \le j \le 7. \end{array}$$

The compression function

The compression function φ processes a 512-bit chaining value h and a 512-bit message block m and outputs the updated chaining value g. Both the chaining value and the message blocks are organised as 8×4 arrays of 16-bit elements.

$$\varphi : \mathrm{GF}(2^{16})^{8\times 4} \times \mathrm{GF}(2^{16})^{8\times 4} \to \mathrm{GF}(2^{16})^{8\times 4} : (h, m) \mapsto g = \varphi(h, m)$$

The updated chaining variable is computed as follows.

1. Initialize the state a: The first 4 columns of a are set h, the last 4 columns of a to m.

2. Apply 12 iterations of the round transformation, which consists of the sequence $\gamma, \theta, \tau, \sigma^r$:

$$b = \left(\bigcirc_{1}^{r=12} \left(\sigma^r \circ \tau \circ \theta \circ \gamma\right)\right)(a).$$

3. Truncate and add the feed-forward: XOR the input chaining value h with the first four columns of b to produce the output chaining value g.

The compression function is depicted in Figure 4.3.

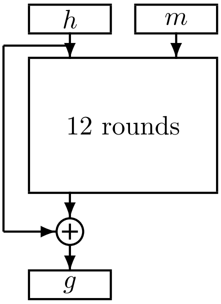


Figure 4.3: $\varphi(h, m)$, the compression function of Whirlwind.

For the detailed specification of how the compression function is iterated in the hash function, we refer to the paper [11].

4.4 Security analysis

In this section, we provide security arguments for Whirlwind by subjecting it to the most important contemporary methods of cryptanalysis and investigating to which extent they apply to Whirlwind.

4.4.1 On basic differential attacks

The design of Whirlwind follows the Wide Trail design strategy [57]. By Theorem 2 of [55], the linear maps θ and τ ensure that a differential trail over R rounds contains at least $\lfloor R/4 \rfloor \mathcal{B}^2$ active S-boxes, where $\mathcal{B}$ is the differential branch number of θ, i.e. 9. In this way, an upper bound of the probability of any differential trail Q through 4 rounds of Whirlwind can be obtained as follows:

$$\mathrm{DP}(Q) \leq \left(2^{-14}\right)^{\mathcal{B}^2} = 2^{-1134}.$$

Given the fact the Whirlwind has a state size of 1024 bits, only a negligible fraction of pairs follows any given differential trail over four rounds or more.

Under the assumption that rare events are hard to reproduce, finding actual conforming pairs for these trails involves a comprehensive computational effort.

4.4.2 Related differentials and algebraic attacks

Without the feedforward, the compression function of Whirlwind can be considered as a block cipher with zero round keys. In this setting, the bound obtained in the previous section can be misleading if the distribution of differential probabilities for the all-zero key significantly deviates from the mean, the EDP.

In the case of AES, Daemen and Rijmen [59] demonstrated that for AES reduced to four rounds, there is a significant fraction of differential trails, called *plateau trails*, for which the distribution of $\mathrm{DP}[k](Q)$ has a large variance, potentially leading to the above-mentioned problem.

The existence of these plateau trails was shown to be caused by the presence of so-called *related differentials* in the linear diffusion layer [61]. Let L be a linear function $L : (\mathrm{GF}(2^n))^{4t} \to (\mathrm{GF}(2^n))^{4t}$, and let $a, b \in (\mathrm{GF}(2^n))^{4t}$. Then the differentials $(a, L(a)), (b, L(b)), (a+b, L(a)+L(b))$ are *related differentials* [61] over L if and only if

$$a_j b_j (a_j + b_j) = 0, \qquad \forall j$$

$$L(a)|_j L(b)|_j \left(L(a)|_j + L(b)|_j\right) = 0, \qquad \forall j \ .$$

It can be proved that related differentials are unavoidable in diffusion layers constructed from $4t \times 4t$ circulant matrices:

Theorem 4.4. *A linear map*

$$L : (GF(2^n))^{4t} \to (GF(2^n))^{4t} : \; x \mapsto L(x) = xL$$

with L a circulant matrix of dimensions $4t \times 4t$, has related differentials.

The proof can be found in [11].

Therefore, any choice of a 8×8 circulant matrix for θ leads to the existence of related differentials. For this reason, Whirlwind uses dyadic matrices instead.

4.4.3 Rebound attacks

In this section, we analyse the applicability of the rebound attack to Whirlwind. First, we discuss some preliminaries specific to rebound attacks on Whirlwind.

Finding a match for γ. The match-in-the-middle step of the inbound phase is commonly making use of a precomputed table to determine values actually following the chosen differential through the S-boxes. For each a and b, this table contains the solutions to the equation $s(x) \oplus s(x \oplus a) = b$. In case of the inversion mapping, either two or four solutions exist for each possible differential (a, b) and the probability that a particular differential exists is about $1/2$.

For Whirlwind's 16-bit S-box, the size of this table is 2^{37} bits, which is not always practical. However, the storage requirement can be lowered without significant increase in computational cost by using a lookup table of $2^{16} \times 2^{16}$ bits just specifying whether a particular differential exists or not. Once a differential match for all S-boxes has been found, only the solutions for the up to 64 different S-box differentials that are actually present have to be calculated. Doing this by brute force takes 2^{22} time, which is negligible since this immediately yields 2^{64} combinations of the two solutions found per individual S-box. In total, 2^{64} values of the state following the differential are expected to be found in $2^{64} + 2^{22}$ time, so the average cost for one match is still about 1.

In comparison with the designs using 8-bit inversion S-boxes, it is interesting to note that merely doubling the size of an individual S-box does not contribute significantly to the average cost of the match-in-the-middle step.

Propagation characteristics of θ. As stated before, the probability of the outbound phase essentially depends on the propagation of truncated differentials through θ. Since each λ_i is an MDS mapping, the sum of active and inputs and outputs is at least equal to 9. Any admissible transition of a active inputs and b active outputs (denoted $a \to b$) in fixed positions requiring b_0 of the b components to be zero occurs with a probability of about 2^{-4b_0}. Consequently, the probability of a $a \to b$ transition of θ with b_0 zero components in fixed positions is lower bounded by 2^{-16b_0}.

The basic differential trail

In what follows, we are using the description of the round function where the explicit calculation of τ is replaced by alternatingly applying θ to the rows (denoted θ_R) and the columns (denoted θ_C). Also note that in terms of differences, σ^r can be neglected and is therefore omitted from the sequence of operations.

In order to obtain a collision for the compression function of Whirlwind, we have to find a differential mapping differences only in the left half of the state (i.e., differences in the message) to a zero difference in the left half of the state (i.e., the output chaining value).

The basic trail of truncated differentials has the following pattern of active S-boxes which is also illustrated in Figure 4.4:

$$4 \xrightarrow{r_1} 8 \xrightarrow{r_2} 64 \xrightarrow{r_3} 8 \xrightarrow{r_4} 8. \tag{4.10}$$

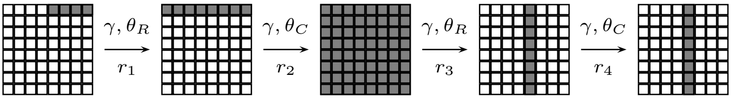


Figure 4.4: Basic truncated differential trail covering 4 rounds.

While this trail does not have the minimum number of active S-boxes according to the wide trail strategy, it minimizes the cost of the outbound phases of the attacks.

Semi-free-start collision on 4.5 rounds

The differential trail (4.10) can be directly used in a rebound attack to obtain a semi-free-start collision for Whirlwind reduced to 4.5 rounds. The attack (see also Figure 4.5) goes as follows.

Inbound phase. The inbound phase of the attack covers the expensive fully active state in the middle of the trail. First, we choose a random difference for the 8 cells of the first row at the input of θ_C in the second round. By the MDS property, we obtain a fully active state at the beginning of round 3. Analogously, we choose another random difference of 8 cells for the fifth column

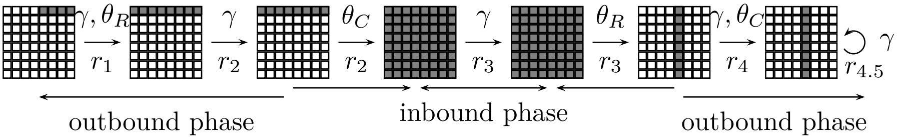


Figure 4.5: The collision attack on 4.5 rounds.

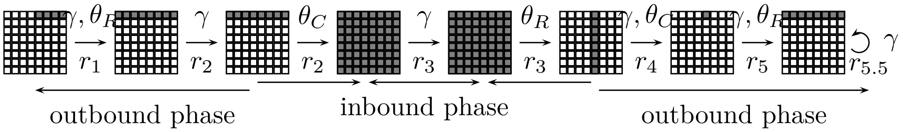


Figure 4.6: The near-collision attack on 5.5 rounds.

at the output of θ_R in round 3 and propagate backwards to obtain another fully active state at the output of γ in round 3.

Using the procedure described above, now a match-in-the-middle is performed to obtain values at the input and output of the S-box layer in round 3 matching the differential. After trying about 2^{64} differences, we can expect to find one existing differential and 2^{64} conforming values.

Outbound phase. The differences and state values obtained in the previous step are now propagated outwards through the γ layers. From now on, we require those to follow certain truncated differential trails. In the backward direction, we require θ_R of round 1 to propagate the 8 active cells of the first row to 4 active cells in the right half of the row. The probability of this is lower bounded by 2^{-64}, since four out of eight cells in specific positions must have a zero difference. In the forward direction, θ_C needs to propagate a fully active column into a fully active column, which happens with a probability of at least $1 - \left(\sum_{i=1}^{7} 2^{-16i}\right)$, which is about 1. At the end, half a round (consisting of γ) can be appended for free since this does not change the activity pattern of the truncated differential.

Summarizing, we need to fulfill one $8 \to 4$ transition in the backward direction. The probability of the outbound phase is thus 2^{-64} so that the 2^{64} values provided by 2^{64} iterations of the inbound phase are just sufficient to get one pair following the trail. A semi-free-start collision for 4.5 rounds can hence be found with complexity 2^{64}.

Semi-free-start near-collision on 5.5 rounds

The semi-free-start collision attack on 4.5 rounds of Whirlwind can be extended to a semi-free-start near-collision attack on 5.5 rounds (see Figure 4.6). Instead of the $8 \to 8$ transition in θ_C of round 4, we require a propagation of 8 to 1 active S-box in the first row. After θ_R in round 5, this expands to a fully active first row with probability 1 due to the MDS property. Since the output chaining value consists of the left half of the state, we obtain a near-collision on 448 of the 512 bits. The inbound phase and the backwards propagation part of the

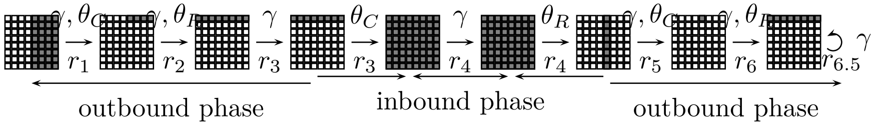


Figure 4.7: The near-collision attack on 6.5 rounds.

outbound phase are exactly the same as in the attack on 4.5 rounds, but the probability of the forwards propagation part decreases: It is now lower bounded by 2^{-112} since we require seven out of eight cells to have a zero difference.

In total, the outbound phase has a probability of 2^{-176} and the complexity of obtaining a near-collision pair for 5.5 rounds is 2^{176}, which is already quite close to a generic birthday attack on 448 bits.

Extension to 5.5/6.5 rounds

Both the collision and near-collision attacks previously described can be extended by prepending one round. In this round, θ is operating on the columns, so that the four active cells in the right half of the first row propagate to four fully active columns in the backwards part of the outbound phase with probability 1. The complexities of the previous attacks hence do not change when extended by one round at the beginning. Note however that such attacks only apply when starting at even round numbers, since we require a zero difference in the left half of the state, a constraint which is violated by having θ operate on rows in the first round. The near-collision attack on 6.5 rounds is illustrated in Figure 4.7.

Extensions to more rounds

Rebound attacks on other wide-trail hash functions have basically used two strategies to extend attacks beyond the basic trail. Either a key or message schedule was used to afford more fully active states by exploiting the freedom available from there (e.g. Whirlpool [122]) or multiple independent inbound phases were efficiently connected by exploiting insufficient diffusion, for instance between parallel states (e.g. Lane [119]).

Both strategies seem inapplicable to Whirlwind due to the absence of influence on intermediate rounds via a key schedule and due to the fact that diffusion is performed according to the wide trail strategy on the whole state and not only on less interconnected parts of it.

4.5 Normal basis decomposition for efficient implementations

Whirlwind makes use of a normal basis tower field decomposition to implement its confusion and diffusion layers.

4.5.1 Properties of normal bases in finite fields

A *normal basis* is constructed by choosing an element $v \in \mathrm{GF}(2^{mp})$ and setting $v^{2^{mj}}, \quad 0 \leq j \leq p-1$. Representing a finite field using normal bases leads to a number of important properties.

Property 4.5 ([111]). *If the elements of the finite field GF*(2^{mp}) *are represented by p-dimensional vectors over GF*(2^m) *using a normal basis, then raising an element to the power* 2^m *corresponds to rotating the coordinates of the element by one position.*

Property 4.6 ([130]). *If the elements of the finite field GF*(2^{mp}) *are represented in a normal basis, then any power map power*(x) *is rotation invariant:*

$$rot(power(x)) = power(rot(x)).$$

We make use of another property of normal bases.

Property 4.7 ([140]). *If* v *is a normal element of GF*(q^{mp}) *with respect to (w.r.t.) GF*(q) *then* $w = Tr_{GF(q^{mp})/GF(q^m)}(v)$ *is a normal element of GF*(q^m) *w.r.t. GF*(q)*.*

Proof. Recall that $w = \mathrm{Tr}_{\mathrm{GF}(q^{mp})/\mathrm{GF}(q^m)}(v) = \sum_{i=0}^{p-1} v^{q^{mi}}$. We will show that the conjugates w^{q^j} $(j = 0 \ldots p-1)$ are linearly independent. Indeed

$$w^{q^j} = \left(\sum_{i=0}^{p-1} v^{q^{mi}} \right)^{q^j} = \sum_{i=0}^{p-1} (v^{q^{mi}})^{q^j} = \sum_{i=0}^{p-1} v^{q^{mi+j}}$$

so each of the conjugates of w is a sum of p different conjugates of v, and all the mp conjugates of v appear exactly once in the m sums of p summands each. Since v was chosen to be a normal element of $\mathrm{GF}(q^{mp})$ over $\mathrm{GF}(q)$, the conjugates of v are linearly independent over $\mathrm{GF}(q)$. Consequently, w is a normal element of $\mathrm{GF}(q^m)$ w.r.t. $\mathrm{GF}(q)$. □

4.5.2 Normal bases in Whirlwind

Since in our case, we are dealing with a decomposition of $\mathrm{GF}(2^{16})$ to $\mathrm{GF}(2^8)$ to $\mathrm{GF}(2^4)$, we consider the case $p = 4$. Let v_2 be an element of $\mathrm{GF}(2^{2m})$, such that $\{v_2, v_2^{\ell_1}\}$ with $\ell_1 = 2^m$ is a normal basis of $\mathrm{GF}(2^{2m})$ over $\mathrm{GF}(2^m)$. Set

- $q_2 \overset{\text{def}}{=} \mathrm{Tr}_{\mathrm{GF}(2^{2m})/\mathrm{GF}(2^m)}(v_2) = v_2 + v_2^{\ell_1}$, so $q_2(\neq 0) \in \mathrm{GF}(2^m)$ and
- $g_2 \overset{\text{def}}{=} q_2^{-1}v_2^2 + v_2 = q_2^{-1}v_2^{\ell_1+1}$, i.e. $g_2(\neq 0) \in \mathrm{GF}(2^m)$.

Let v_4 be an element of $\mathrm{GF}(2^{4m})$ such that $\{v_4, v_4^{\ell_2}\}$ with $\ell_2 = 2^{2m}$ be a normal basis of $\mathrm{GF}(2^{4m})$ over $\mathrm{GF}(2^{2m})$. Define

- $q_4 \overset{\text{def}}{=} \mathrm{Tr}_{\mathrm{GF}(2^{4m})/\mathrm{GF}(2^{2m})}(v_4) = v_4 + v_4^{\ell_2}$, so $q_4(\neq 0) \in \mathrm{GF}(2^{2m})$ and
- $g_4 \overset{\text{def}}{=} q_4^{-1}v_4^2 + v_4 = q_4^{-1}v_4^{\ell_2+1}$, i.e. $g_4 \in \mathrm{GF}(2^{2m})$.

The elements v_2 and v_4 can be chosen independently or we can choose a normal element $v_4 \in \mathrm{GF}(2^{4m})$ and then derive $v_2 = \mathrm{Tr}_{\mathrm{GF}(2^{4m})/\mathrm{GF}(2^{2m})}(v) \in \mathrm{GF}(2^{2m})$. Using Property 4.7 it follows that v_2 is a normal element in $\mathrm{GF}(2^{2m})$. Note that in this case we have $q_4 = v_2$ and $q_2 = \mathrm{Tr}_{\mathrm{GF}(2^{2m})/\mathrm{GF}(2^m)}(v_2) = \mathrm{Tr}_{\mathrm{GF}(2^{4m})/\mathrm{GF}(2^m)}(v)$ (using trace transitivity) and hence $q_4, q_2 \neq 1$ (Property 4.7 implies that q_2 is a normal element in $\mathrm{GF}(2^m)$).

4.5.3 Multiplication and inversion using normal bases

Analogously to [130], using a normal basis decomposition leads to simple and efficiently implementable formulas for products and inverses of elements.

Let (a, b) and (c, d) be the coordinates of two elements of $\mathrm{GF}(2^{2m})$. Then the coordinates of the product are given by the following formula [11]:

$$(e, f) = (a, b) \times (c, d) \quad \Leftrightarrow \quad \left\{ \begin{array}{lcl} e & = & (a+b)(c+d)g_2 + q_2ac \\ f & = & (a+b)(c+d)g_2 + q_2bd \end{array} \right.$$

In an analogous fashion, let now (a, b) and (c, d) be the coordinates of two elements of $\mathrm{GF}(2^{4m})$. The formula for the product in this case then given by:

$$(e, f) = (a, b) \times (c, d) \quad \Leftrightarrow \quad \left\{ \begin{array}{lcl} e & = & (a+b)(c+d)g_4 + q_4ac \\ f & = & (a+b)(c+d)g_4 + q_4bd \end{array} \right.$$

For the inversion, a similar formula can be derived [11]. With (a, b) the coordinates of an element of $\mathrm{GF}(2^{2m})$, the coordinates of the inverse element are given by the following formula:

$$(c, d) = (a, b)^{-1} \quad \Leftrightarrow \quad \left\{ \begin{array}{lcl} c & = & ((a+b)^2 g_2 q_2 + q_2^2 ab)^{-1} b \\ d & = & ((a+b)^2 g_2 q_2 + q_2^2 ab)^{-1} a \end{array} \right.$$

Analogously let (a, b) be the coordinates of an element of $\mathrm{GF}(2^{4m})$. The inverse can then be computed by the following formula:

$$(c, d) = (a, b)^{-1} \quad \Leftrightarrow \quad \left\{ \begin{array}{lcl} c & = & ((a+b)^2 g_4 q_4 + q_4^2 ab)^{-1} b \\ d & = & ((a+b)^2 g_4 q_4 + q_4^2 ab)^{-1} a \end{array} \right. \tag{4.11}$$

In this way we can decompose the inversion map from $\mathrm{GF}(2^{4m}) \rightarrow \mathrm{GF}(2^{2m}) \rightarrow \mathrm{GF}(2^m)$. For the Whirlwind proposal $m = 4$ so we get the decomposition $\mathrm{GF}(2^{16}) \rightarrow \mathrm{GF}(2^8) \rightarrow \mathrm{GF}(2^4)$.

4.5.4 Basis choice for Whirlwind

In Whirlwind, $\mathrm{GF}(2^{16})$ is recursively decomposed into smaller subfields according to Section 4.5.2 by choosing a normal element of $\mathrm{GF}(2^{16})$ and using the traces of this element into the subfields to construct the normal bases. This decomposition is employed uniformly at each level, so that all individual field extensions have degree two:

$$\mathrm{GF}(2^{16}) \rightarrow \mathrm{GF}(2^8) \rightarrow \mathrm{GF}(2^4) \rightarrow \mathrm{GF}(2^2) \rightarrow \mathrm{GF}(2).$$

In order to describe our choice for the normal bases unambiguously, we use the field representations given in Table 4.1(a) as a reference. The normal bases used in Whirlwind, together with the elements q_i, g_i used for the field arithmetic are summarised in Table 4.1(b).

4.5.5 Practical implementation considerations

The square shape of its state and the fact that it is designed according to the wide trail strategy imply that Whirlwind can be implemented using approaches developed for the AES [57] and especially Whirlpool [12]. We briefly recall the techniques here and discuss how to apply them to Whirlwind.

Table 4.1: Overview of the normal bases decomposition in Whirlwind.

(a) Field representations used for unique reference.

Field	Defining polynomial
$\mathrm{GF}(2^{16}) \cong \mathrm{GF}(2)(z)$	$X^{16} + X^5 + X^3 + X^2 + 1$
$\mathrm{GF}(2^8) \cong \mathrm{GF}(2)(y)$	$X^8 + X^4 + X^3 + X^2 + 1$
$\mathrm{GF}(2^4) \cong \mathrm{GF}(2)(x)$	$X^4 + X + 1$
$\mathrm{GF}(2^2) \cong \mathrm{GF}(2)(w)$	$X^2 + X + 1$

(b) Normal basis decomposition.

Field	Normal element	q_i	g_i
$\mathrm{GF}(2^{16})$	$v_4 = z^{101}$	—	—
basis over $\mathrm{GF}(2^8)$: $\{v_4, v_4^{256}\}$			
$\mathrm{GF}(2^8)$	$v_2 = y^{101}$	$q_4 = v_2$	$g_4 = 1$
basis over $\mathrm{GF}(2^4)$: $\{v_2, v_2^{16}\}$			
$\mathrm{GF}(2^4)$	$v_1 = x^7$	$q_2 = v_1$	$g_2 = x^4$
basis over $\mathrm{GF}(2^2)$: $\{v_1, v_1^4\}$			
$\mathrm{GF}(2^2)$	$v_0 = w$	$q_1 = v_0$	$g_1 = 1$
basis over $\mathrm{GF}(2)$: $\{v_0, v_0^2\}$			
$\mathrm{GF}(2)$			

Software implementation on standard microprocessors

The standard approach to implement the linear layer θ is to use lookup tables containing all scalar products with each of the rows, where the S-box application is integrated on the scalars.

For Whirlwind, θ consists of the parallel application of four matrix multiplications over $\mathrm{GF}(2^4)^{8\times 8}$ on the rows of the state. Combining their contribution to one output row into a single table, both γ and θ can be implemented with eight table lookups. Denote by M_{ik} the k-th row of the dyadic matrices M_0 and M_1 and define the table $T_k, 0 \leq k \leq 7$ by

$$T_k\left[\begin{pmatrix} x_{0,0} & x_{0,1} \\ x_{1,0} & x_{1,1} \end{pmatrix}\right] = s\left[\begin{pmatrix} x_{0,0} & x_{0,1} \\ x_{1,0} & x_{1,1} \end{pmatrix}\right] \cdot \left(\begin{pmatrix} M_{0k,0} & M_{1k,0} \\ M_{1k,0} & M_{0k,0} \end{pmatrix}, \ldots, \begin{pmatrix} M_{0k,7} & M_{1k,7} \\ M_{1k,7} & M_{0k,7} \end{pmatrix}\right). \tag{4.12}$$

Each row b_i of $b = (\theta \circ \gamma)(a)$ is then equal to

$$b_i = \bigoplus_{k=0}^{7} T_k[a_{i,k}]. \tag{4.13}$$

Due to the larger S-box, the memory requirements of each table is $2^{16} \cdot 128$ bits, i.e. one megabyte.

Impact of τ. In practice, each of the 2×2 submatrices comprising one element of the state will be identified by the vector $(x_{0,0}, x_{0,1}, x_{1,0}, x_{1,1})$ and both a row of the state and the entries of the T_k will be organized in machine-sized words. For Whirlwind, each row then comprises two 64-bit or four 32-bit words. However, this layout of the state implies that operations across rows become significantly more efficient than operations across columns.

In designs using cyclic shifts to diffuse across second dimension, this operation can be implemented by a simple reordering of indices in (4.13), so that only row operations need to be performed. This is however not possible for a matrix transposition.

As noted in Section 4.3.1, the calculation of τ can be avoided by letting θ operate on rows and columns of the state alternatingly. The normal description of Whirlwind's round transformation:

$$b = (\sigma^r \circ \tau \circ \theta \circ \gamma)(a)$$

then turns into

$$b = \begin{cases} \left((\sigma^r)^T \circ \theta \quad \circ \gamma\right)(a) & \text{for odd } r, \\ \left(\ \sigma^r \quad \circ \theta_C \circ \gamma\right)(a) & \text{for even } r \end{cases}$$

where $(\sigma^r)^T$ denotes the application of the transpose of the r-th round constant and θ_C is θ operating on the columns.

In a state layout representing a row using machine words, the computation of θ_C still involves one matrix transposition, but the tables for θ can be reused and the number of transpositions is reduced to $R/2$, with R being the total number of rounds.

The transpose can for example be efficiently implemented using a recursive decomposition of the 8×8 transpose into transpositions of 4×4 and 2×2 matrices using SIMD instructions such as shuffle and unpack in Intel's SSE instruction set.

Table 4.2: Performance figures for software implementations of Whirlwind on Intel processors. All numbers are given in cycles per byte (cpb).

	Xeon E5540 2.53 GHz, 8 MB L3 cache	Core 2 X9650 3 GHz, 6 MB L2 cache	Xeon E5335 2 GHz, 4 MB L2 cache
Large lookup tables	151.31 cpb	210.32 cpb	217.86 cpb
Compressed inverse	129.43 cpb	135.61 cpb	207.74 cpb
Medium lookup tables	99.58 cpb	124.35 cpb	145.18 cpb
Bitsliced (2 blocks)	63.22 cpb	63.14 cpb	67.79 cpb
Whirlpool	54.60 cpb	38.15 cpb	56.31 cpb

Performance of the table-based approach. While arguably being an efficient method to implement Whirlwind in theory, the memory required to store all eight lookup tables is 8 MB, which exceeds the L2 cache size available to a single core in most contemporary processors. As a result, parts of the table are constantly cached in and out, resulting in a significant performance penalty. As indicated in Table 4.2, the performance greatly varies with the size of the L2/L3 cache. A significant speedup is expected on CPUs with 12 or 16 MB of cache such as Intel's Xeon 7400 series.

Using smaller tables. In order to avoid L2/L3 cache pressure, the Whirlwind round transformation can also be implemented using smaller lookup tables. In this cases, γ has to be implemented by separate table lookups requiring a table of $2^{16} \cdot 2$ bytes. Since the λ_i are applied independently, the approach described in equation (4.12) can be generalized to lookup tables for one, two or four λ_i mapping scalars of 4, 8 or 16 bits to complete 128-bit rows with the individual contributions shifted to the final location. Then an output row can be obtained by XOR-ing 4, 2 or 1 partial rows together 8 times. The best trade-off for contemporary machines is offered by combining two λ_i per table. In total, this implementation needs $2^{17} + 2^{16}$ bytes (192 KB) of lookup tables, which easily fits in most L2 caches.

As seen in Table 4.2, this implementation ("medium lookup tables"), albeit needing three times as many lookups per round transformation as in the big tables approach, improves performance especially on CPUs with smaller cache size. The speedup this implementation receives on the machines with bigger cache sizes are explained by them also featuring more recent microarchitectures.

Using the symmetry of the inverse. The use of normal bases in Whirlwind permits another implementation variant based on the following symmetry property: If $(a,b)^{-1} = (c,d)$, then $(b,a)^{-1} = (d,c)$ and $(a,a)^{-1} = (c,c)$. This can be directly verified by equation (4.11) and suggests the following procedure for computing the inverse of (a,b) in $\mathrm{GF}(2^{4m})$:

$$(a,b)^{-1} = \begin{cases} S[a,b] & \text{if } a < b, \\ (S[a,a], S[a,a]) & \text{if } a = b, \\ S[b,a] & \text{if } a > b; \end{cases}$$

with a table S storing only the inverses (c,d) of (a,b) with $a < b$ and the value c for the inverse of $(a,a)^{-1} = (c,c)$. This table then only requires $2^{2m} \cdot (2^{2m} - 1)/2 \cdot 4m + 2m \cdot 2^{2m} = 2m \cdot 2^{4m}$ bits of memory, which is half the size of the full lookup table. For Whirlwind, $m = 4$, so this implies a table of 64 KB instead of 128 KB.

However, this reduction comes at the expense of conditional processing. So while this technique can also be used to halve the size of the tables described by (4.12), the cost of the additional processing practically compensates for this, as seen in Table 4.2. On the other hand, table-based implementations on platforms where memory is the primary concern will benefit from the reduction provided by this technique (see also Section 4.5.5).

Bitslicing. In order to obtain a constant-time implementation resistant to side-channel attacks, Whirlwind can also be implemented in a bitsliced manner. Similar to the recent very fast bitsliced implementation of the AES [99], the normal basis decomposition of the field arithmetic presented in Section 4.5 leads to compact formulations on the level of individual bits. The current implementation is using Intel's SSE3 instruction set and processes two blocks of two independent hashing operations in parallel to fully utilize the register width. Improving this implementation and deriving a representation for single block hashing that still leads to efficient computations is an interesting target for future work.

Embedded platforms and hardware

At the moment, there are no implementations for 8-bit processors or in hardware. However, by making use of the tower field decomposition employed in Whirlwind, the implementation techniques described in [130] can be applied. Also, we expect the compactness estimations from this paper to carry over proportionally, resulting in competitive implementations for resource-constrained platforms.

Memory requirements on 8-bit platforms. For implementations on 8-bit microcontrollers, the requirements in terms of RAM and ROM are generally a far greater concern than execution speed. We estimated those requirements for Whirlwind according to the criteria mentioned in [89], which in particular implies that the IV is stored in ROM and that the memory used for the message block is considered external to the hash algorithm and hence not taken into account.

An implementation of Whirlwind needs 512 bits of ROM for the IV and at least 512 bits of RAM to store the previous chaining value for the feed-forward and 1024 bits of RAM for the internal state. The round constants can either be implemented using 768 bits of ROM or via a counter using 8 bits of RAM. As described in Section 4.5.3, the finite field arithmetic can be recursively decomposed down to binary operations. While this comes at a performance penalty on 8-bit platforms, it eliminates the need for lookup tables. Alternatively, the technique from Section 4.5.5 can be used to speed up the implementation at the cost of 2^{16} bytes of ROM for a lookup table of the compressed inverse. Combined approaches are possible, for instance using the normal basis arithmetic for the first level of decomposition and then employing lookup tables for the smaller subfields.

Summarising, we estimate that Whirlwind can be implemented on 8-bit platforms using either 192 bytes of RAM and 160 bytes of ROM, or 193 bytes of RAM and 64 bytes of ROM. Compared to the SHA-3 candidates analyzed in [89], this places Whirlwind in the “Middle” class of algorithms with regard to memory requirements, for instance being significantly smaller than the Round 2 candidates ECHO and SIMD, comparable to BMW, Shabal and the finalist Keccak, and larger than the finalist BLAKE and the Round 2 candidates Hamsi and Luffa.

4.6 Grøstl-0: Attacking a rebound-resistant hash function

Since the rebound attack was discovered by its designers [122], the hash function Grøstl[75] is designed to resist it to a large extent. The evaluation to which extent rebound techniques can provide attacks on reduced-round versions of Grøstl is the subject of the remainder of this chapter.

The Grøstl hash function was selected as a finalist of the NIST SHA-3 competition. Since the designers tweaked the specification before the final round of the competition, partially also in response to the attacks presented

Target	Digest Size	Rounds	Time	Memory	Type	Reference
hash function (10 rounds)	224, 256	4	2^{64}	2^{64}	coll.	[123]
	256	5	2^{48}	2^{32}	coll.	Sect. 4.7.2
	256	6	2^{112}	2^{32}	coll.	Sect. 4.7.2
	224	5	2^{48}	2^{32}	coll.	Sect. 4.7.2
compression function (512-bit CV)	224, 256	10	$2^{170.7}$	$2^{170.7}$	FSC	[74]
	256	7	2^{120}	2^{64}	SFSC	[77, 123]
	224, 256	7	2^{80}	2^{64}	SFSC	Sect. 4.7.2
	224, 256	8	2^{192}	2^{64}	SFSC	Sect. 4.7.2
hash function (14 rounds)	384, 512	5	2^{176}	2^{64}	coll.	[123]
	512	6	2^{183}	2^{64}	coll.	Sect. 4.7.3
compression function (1024-bit CV)	384, 512	14	$2^{341.3}$	$2^{341.3}$	FSC	[74]
	384, 512	7	2^{152}	2^{64}	SFSC	[123]
	384, 512	7	2^{152}	2^{56}	SFSC	[20]
	384, 512	8	2^{224}	2^{96}	SFSC	Sect. 4.7.3
	384, 512	9	2^{504}	2^{128}	SFSC	Sect. 4.7.3
permutation	224, 256	7	2^{55}	−−	dist.	[121]
	224, 256	7	2^{19}	−−	dist.	Sect. 4.8.2
	224, 256	8	2^{112}	2^{64}	dist.	[77]]
	224, 256	8	2^{48}	28	dist.	[149]
	224, 256	8	2^{64}	2^{64}	dist.	Sect. 4.8.1

Table 4.3: Summary of results for Grøstl-0. “coll.”, “FSC”, “SFSC”, and “dist.” are abbreviations for collision, free-start collision, semi-free-start collision, and distinguisher, respectively.

here, there are two versions of Grøstl: the original version for round 1 and 2 [74], called Grøstl-0, and the tweaked version for the final round [75]. Our target is Grøstl-0.

Grøstl-0 has been extensively cryptanalysed [77, 121–123, 141, 149]. We give a short overview of cryptanalysis results on Grøstl-0 together with an overview of our results presented in the next sections in Table 4.3. These results have been published jointly with Kota Ideguchi and Bart Preneel in [90] and [91].

4.6.1 Description of Grøstl-0

We give a short explanation of the specification of Grøstl-0. For a detailed explanation, we refer to the original paper [74]. Grøstl-0 is an iterated hash

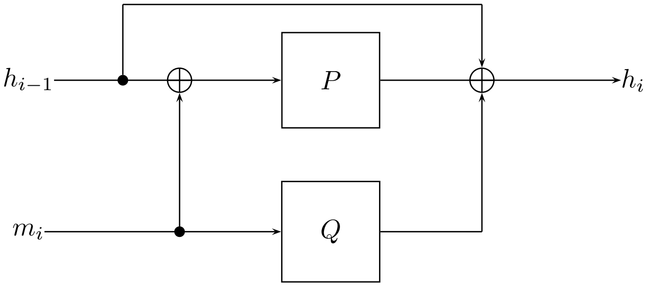


Figure 4.8: The compression function of Grøstl

function with an SPN structure following the wide trail design strategy. There are two variants of Grøstl-0: one for hash lengths of 224 and 256 bits, and one for 384 and 512 bits.

The size of the chaining values are 512-bit for Grøstl-0-224 and -256; they are stored as an 8 by 8 matrix whose elements are bytes. The compression function takes a 512-bit chaining value (CV) and a 512-bit message block as inputs and generates a new 512-bit CV. The compression function uses two 512-bit permutations P and Q. The compression function $\mathcal{G}$ is defined as follows (see also Figure 4.8):

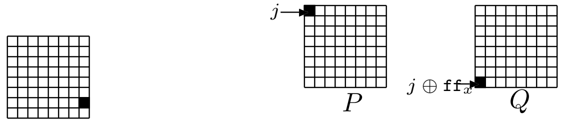
$$h_i = \mathcal{G}(h_{i-1}, m_i) \overset{\text{def}}{=} P(h_{i-1} \oplus m_i) \oplus Q(m_i) \oplus h_{i-1}.$$

A message is padded and divided into 512-bit message blocks $m_1, \dots, m_b$ and processed as follows to generate a hash value h:

$$\begin{aligned} h_0 &= IV, \\ h_i &= \mathcal{G}(h_{i-1}, m_i) \qquad i = 1, \dots, b \\ h = &= P(h_b) \oplus h_b. \end{aligned}$$

The initial value IV is the binary representation of the hash length. For example, the IV of Grøstl-0-256 is IV = $\texttt{00} \ \ldots \texttt{0100}_x$, its matrix representation is depicted in Figure 4.9a.

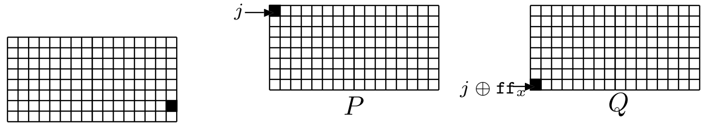
We denote by MB_n the MixBytes-n operation used in the Grøstl-0-n permutations; similarly SB_n denotes SubBytes-n, ShB_n denotes ShiftBytes-n , and AC_n denotes AddConstants-n. Here SB_n is a non-linear transformation using the AES S-box. ShB_n consists of a byte-wise cyclic shift of rows. In the case of $n = 256$, the i-th row is cyclically rotated to the left by i bytes. MB_n



(a) IV of Grøstl-0-256

(b) AddConstant operation of Grøstl-0-256

Figure 4.9: IV and AddConstant in Grøstl-0-256



(a) IV of Grøstl-0-512

(b) AddConstant operation of Grøstl-0-512

Figure 4.10: IV and AddConstant in Grøstl-0-512

is a matrix multiplication, where a constant MDS matrix is multiplied from the left to all columns independently. AC_n of P XORs the round number to the first bytes of the internal state. AC_n of Q xors the negation of the round number to the eighth byte of the internal state. These functions are depicted in Figure 4.9b. The permutation P is a wide-trail [56] round function with 10 rounds, where a round consists of $\mathrm{MB}_{256} \circ \mathrm{ShB}_{256} \circ \mathrm{SB}_{256} \circ \mathrm{AC}_{256}$. The permutation Q has the same structure; it uses the same MB_{256}, ShB_{256} and SB_{256} but another AC_{256}.

For Grøstl-0-384 and Grøstl-0-512, the chaining values are 1024-bit in size and are stored as an 8 by 16 byte matrix. The compression function has the same form as Grøstl-0-256, except that the lengths of the permutations P and Q are 1024 bits. Again, the IV is the binary representation of the hash length; the matrix representation of the IV of Grøstl-0-512 is depicted in Figure 4.10a. The number of rounds is 14 and each round consists of MB_{512}, ShB_{512}, SB_{512} and AC_{512}, which are the natural 1024-bit equivalents of their 512-bit counterparts, except for ShB_{512}, which moves each row left cyclically by i bytes except for the last row, which is moved left cyclically by 11 bytes. AC_{512} of P xors the round number to the first bytes of the internal state. AC_{512} of Q xors the negation of the round number to the eighth byte of the internal state. These functions are depicted in Figure 4.10b.

In the remainder of this chapter, we will omit the suffix n of each function, if it is clear from the context.

4.6.2 The internal differential attack

The internal differential attack was introduced by Peyrin [141] for the cryptanalysis of the Grøstl-0 permutation. It is a general technique that can apply to algorithms using parallelly executed components (e.g., permutations) that are very similar. Instead of tracing the propagation of differences of a pair (x, x^*) through the same function F, it traces the propagation of differences of x through two functions F and F^*. Here, similarity is measured in terms of XOR differences.

It is obvious that this approach can be suitable for Grøstl-0, since the difference between the two parallel permutations calls $P(x)$ and $Q(x)$ only lies in the AddConstant operation, which only modifies one byte per round.

The internal differential attack has been used by Peyrin to construct distinguishers for the internal permutation of Grøstl-0 [141]. However, his approach cannot be used to obtain (semi-free-start) collisions on the compression of hash function due to lack of degrees of freedom. In the sequel, we will study how the internal differential attack can be used in collision attacks.

4.6.3 Variants of the rebound attack

In this section, we outline the start-from-the-middle and Super-Sbox variants of the rebound attack. For a detailed explanation, we refer to the original papers [77, 106, 121].

We recall two properties of the MDS matrix used in MB which we will frequently use in the sequel. First, the MDS property implies that the sum of the number of active bytes of the truncated difference at the input and at the output is always at least 9. Furthermore, the property implies that the probability of a transition where the input (output) difference is randomly chosen and the number of active bytes of the output (input) difference is k is about $2^{-8(8-k)}$ [123].

The start-from-the-middle rebound technique

The start-from-the-middle variant of the rebound attack is due to Mendel *et al.* [121]. We outline this technique using a small example with a 4-round path of Grøstl-0-256 depicted in Figure 4.11. We split the path into a controlled ("inbound") phase and an uncontrolled ("outbound") phase. The controlled phase consists of the middle part of the path and comprises the most costly

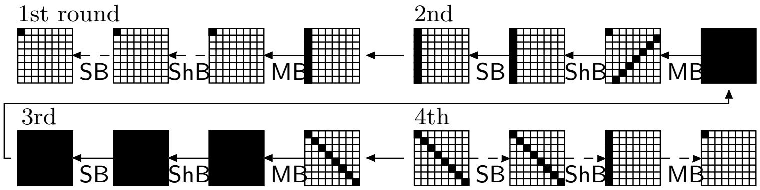


Figure 4.11: Example path for the start-from-the-middle rebound attack.

differential transitions. In Figure 4.11, the controlled phase is shown by solid arrows and the uncontrolled one is shown by dashed arrows.

First, the difference distribution table (DDT) of the Sbox it built; this is a two-dimensional table whose rows and columns are labeled by the input differences and the output differences, respectively, and an element contains the corresponding input pairs. It takes 2^{16} time to construct the table of size 2^{16}. Next, the controlled phase is executed:

1. We fix the input difference of the 4th round SB, then calculate the output difference of the 3rd round SB.
2. We fix the input difference of the 2nd round MB and calculate the input difference of the 3rd round SB. Then, for each column of the output of the 3rd round SB, we determine pairs of values of the column by using the precomputed difference distribution table such that the fixed input and output differences of the 3rd-round SB are complied with. Note that this can be done for each column independently. By repeating this step with different input differences of the 2nd-round MB, we obtain a set of solutions for each column, hence eight sets of solutions.
3. For each of the solutions obtained in the previous step, the differences of the input of the 2nd-round SB are determined. We then select these solutions that cause the difference of the input of the 2nd-round SB to be transformed to a one-byte difference by the 1st-round inverse MB. These are the solutions of the controlled phase. We can repeat the procedure from Step 1 to 3 by changing the input difference of the 4th round SB.

Note that while we described the procedure starting from the end of the 3rd round backwards, following it in the opposite direction is possible as well. In both cases, the average time complexity to find one solution of the controlled phase is equal to one, with negligible memory complexity.

The uncontrolled phase is probabilistic. In the backward direction (from the input of the 1st-round ShB to the beginning of the path), the probability to

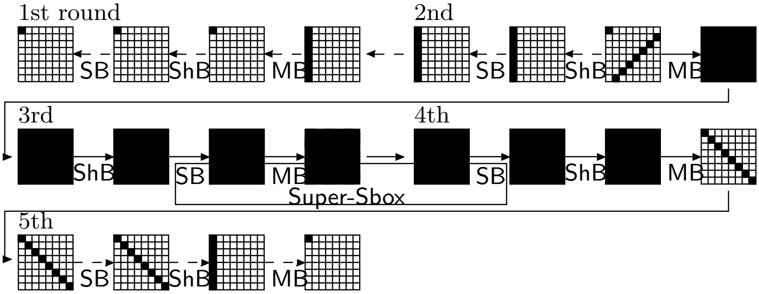


Figure 4.12: Example path for the Super-Sbox rebound attack.

follow the path is almost one. In the forward direction (from the input of the 4th-round SB to the end of the path), it requires 2^{56} solutions of the controlled phase in order to follow a $8 \to 1$ transition for the 4th-round MB. This implies that in total, generating one solution of the whole path takes time 2^{56}.

The available degrees of freedom can be counted based on the method of [141]. For the path in Figure 4.11, we can find about 2^{15} solutions following the whole path. Note that if we consider differential paths between $P(m+IV)$ and $Q(m)$, we do no need to halve the overall degrees of freedom at the middle point. Hence in our case, the degrees of freedom can actually be doubled compared to the method of Peyrin [141].

The Super-Sbox rebound technique

The Super-Sbox rebound technique [77, 106] combines the *super box* concept with the rebound attack. The super box concept [59] considers the S-box layers of two consecutive rounds as one big S-box layer. More precisely, by exchanging the order of the first SB and the first ShB, we have a composition of SB ∘ AC ∘ MB ∘ SB. This composition is considered as eight independent 64-bit to 64-bit Sboxes, which are called super boxes, or Super-Sboxes as in [77]. We explain the Super-Sbox rebound technique by an example in Figure 4.12. Similar to the start-from-the-middle rebound technique, we split the path into a controlled phase and an uncontrolled phase.

For the controlled phase, we follow the following steps:

1. Fixing the input difference of the 2nd round MB, we calculate the input difference of the 3rd round SB. For each Super-Sbox, we compute the output differences of the Super-Sbox for all possible input pairs which

have a fixed input difference and construct tables of the output difference and the corresponding input pairs. This takes time and memory 2^{64}.

2. We fix the output difference of the 4th round MB and calculate the output difference of the 4th round SB.
3. For each Super-Sbox, we search for the pairs of the inputs of the Super-Sbox by using the partial difference distribution table of Step 1 such that the output difference of the 4th-round SB equals the one of Step 2. This can be done for each Super-Sbox independently. In case we do not obtain enough solutions for the controlled phase, steps 2 and 3 can be repeated until the output differences of the 4th round MB are exhausted. After exhausting the differences at Step 2, we can backtrack to Step 1 by changing the input difference of the 2nd round MB.

Although we described the attack proceeding forward from the 2nd round MB, starting in the 4th round and computing backwards is equally possible.

The uncontrolled outbound phase is again probabilistic. Since there are two $8 \rightarrow 1$ transitions (the 1st-round MB and the 5th-round MB), we need $2^{56 \cdot 2}$ solutions of the controlled phase to expect one remaining solution after the outbound phase. Consequently, the time complexity to generate a solution for the whole path is 2^{112}. The memory complexity is 2^{64}. The degrees of freedom can be counted as in the case of the start-from-the-middle rebound attack.

4.7 Collisions for round-reduced Grøstl-0

4.7.1 Attack strategy

We will apply the previously outlined techniques to obtain collision and semi-free-start collision attacks on the reduced-round Grøstl-0 hash function and compression function.

In all our attacks, we consider differential paths between $P(m \oplus IV)$ and $Q(m)$. The key observation is that in Grøstl-0, the permutations P and Q only differ in the AC functions. Furthermore, each AC operation only introduces a two-byte difference per round. This means that we can construct differential paths between P and Q which hold with high probability. This strategy was introduced by [141] to construct distinguishers for the compression function.

In our attacks on the hash function, we construct a message pair in which each padded message has the same number b of blocks with $b \geq 2$. Consider the

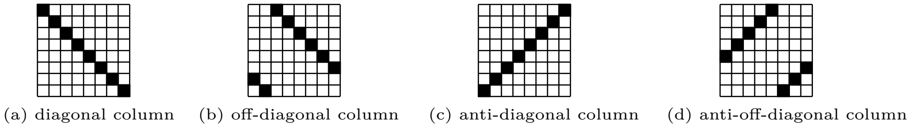

(a) diagonal column (b) off-diagonal column (c) anti-diagonal column (d) anti-off-diagonal column

Figure 4.13: Truncated difference patterns for the analysis of Grøstl.

minimal case $b = 2$. We first find a pair of the first blocks m_1 and m_1' that generates an internal collision of the chaining value after processing the first block:

$$h_i = \mathcal{G}(IV, m_1) = \mathcal{G}(IV, m_1'). \tag{4.14}$$

Then, the same message block m_2 is appended to both blocks such that the padding rule is satisfied. We call the resulting padded messages M and M', respectively. They now generate the same hash value, producing a collision for the hash function. As finding the second message blocks is easy, we can concentrate on finding the first blocks.

For our attacks on the compression function, finding the first blocks is sufficient.

In our analysis, we will make use of certain specific truncated difference patterns in the Grøstl state. A *diagonal column* denotes 8 diagonal bytes, which are aligned to the first column by ShB, and a *off-diagonal column* denotes 8 bytes which are aligned to a column other than the first column by ShB. An *anti-diagonal column* denotes 8 anti-diagonal bytes, which are aligned to the eighth column by inverse ShB, and an *anti-off-diagonal column* denotes 8 bytes which are aligned to a column other than the eighth column by inverse ShB. These terms are illustrated in Figure 4.13.

4.7.2 Attacks on Grøstl-0-256

Collisions for 5-round Grøstl-0-256

We describe a collision attack on the Grøstl-0-256 hash function reduced to 5 rounds. Our differential path between $P(m \oplus IV)$ and $Q(m)$ of the first message block is shown in Figure 4.14. The controlled phase is shown by solid arrows and the uncontrolled phase by dashed arrows. The internal states at the boundaries between the controlled and uncontrolled phases are denoted by A and B, respectively, as indicated in Figure 4.14.

For the controlled (inbound) phase, we start at state A and calculate backwards to state B. Using the start-from-the-middle technique, this can be done with

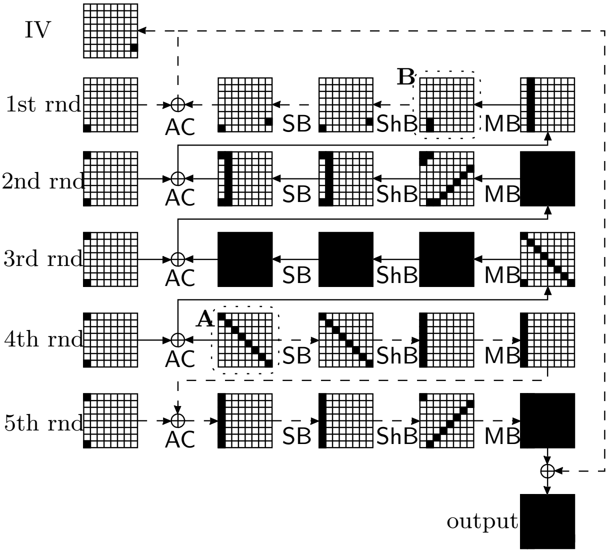


Figure 4.14: Differential path between P and Q of 5-round Grøstl-0-256

average time complexity one. The outbound phase is uncontrolled and has to be followed probabilistically.

In the forward direction (from state A to the end of compression function), the probability to follow the path is almost one. In the backward direction (from state B to the beginning of the compression function), we need 2^{16} repetitions to follow the inverse AC and addition of the IV in the 1st round (two 8-bit conditions). In total, the time complexity to find a message block following the whole path is 2^{16}.

As the differences of the chaining value at the end of the path are determined by the 8-byte difference before the final MB, by the birthday paradox we need to have 2^{32} message blocks following the path in order to obtain a pair (m_1, m_1') whose CVs collide; $P(m_1 \oplus IV) \oplus Q(m_1) \oplus IV = P(m_1' \oplus IV) \oplus Q(m_1') \oplus IV$. Therefore, the total complexity of the attack is $2^{16+32} = 2^{48}$ time and 2^{32} memory.

By counting the degrees of freedom, we observe that we can generate at most 2^{64} message blocks following the path. Because the attack requires 2^{32} message blocks following the path, we have enough degrees of freedom for the purposes of our attack.

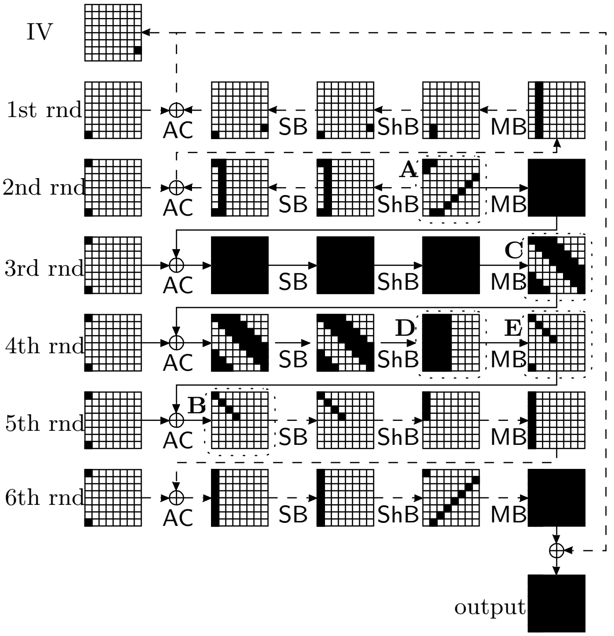


Figure 4.15: Differential path between P and Q of 6-round Grøstl-0-256

Collisions for 6-round Grøstl-0-256

One more round of Grøstl-0-256 can be attacked using the differential path depicted in Figure 4.15.

Again, we denote the internal states at the boundaries between the controlled and uncontrolled phases A and B. Additionally, we denote three more states in the controlled phase by C, D, and E; as indicated in Figure 4.15.

The controlled phase (solid line) proceeds forward from state A to state B. forwards and ends at the state B. Using the start-from-the-middle technique, this can be done with average time complexity one.

For the outbound phase, we have the following probabilistic transitions: In the forward direction (from state B onwards), the probability to follow the path is almost one, while in the backward direction (from state A backwards), it takes 2^{16} repetitions to find a solution following the 2nd round inverse AC, 2^{48} for the 2nd round inverse MB, and 2^{16} for the inverse AC and addition of the IV in the 1st round. With 80 bits of conditions in total, the overall time is about 2^{80}.

Compared to the description of the standard start-from-the-middle algorithm in Section 4.6.3, controlling the transitions from states C to E requires additional steps: In the following, we describe an algorithm to generate 2^{32} solutions of the controlled phase with time and memory 2^{32}, hence still achieving an average cost of one to find one solution.

First, steps 1 and 2 of the start-from-the-middle algorithm are executed normally. We then prepare 2^{32} solutions for each column at the state C independently. Then, we perform the following steps:

1. For each of the solutions obtained in Step 2, we calculate forwards until state D. We now have 2^{32} solutions for each anti-diagonal or anti-off-diagonal column at state D.
2. We pick one solution corresponding to the anti-diagonal column in state D, thereby determining 4 bytes of the anti-diagonal 8-byte difference at the state D. Since the transition pattern of the 4th round MB in the path imposes 28 bytes linear equations on the input difference, the other 28-byte difference at state D is fixed by solving these 28 bytes equations. Then, we can verify whether this 28-byte difference is included in the other sets of the solutions (corresponding to anti-off-diagonal columns). This procedure is then repeated for all 2^{32} elements of the solutions corresponding to the anti-diagonal column.

This algorithm requires 2^{32} time and 2^{32} memory. Since there are $2^{32\cdot 8}$ candidates for solutions at the state D, we obtain $2^{32\cdot 8}/2^{28\cdot 8} = 2^{32}$ solutions at state E by this algorithm. Consequently, we can generate one solution of the controlled phase with average complexity one.

Similar to the 5-round collision attack, we need 2^{32} message blocks following the path in order to obtain a pair (m_1, m_1') with colliding CVs. Hence, the total complexity of the attack is $2^{80+32} = 2^{112}$ time and 2^{32} memory.

By counting the degrees of freedom, we find that we can generate at most 2^{32} message blocks following the path, which is sufficient since our attack requires just 2^{32} message blocks following the path to obtain one collision of CV with high probability.

Semi-free start collisions for 8-round Grøstl-0-224 and -256

We now describe how to obtain semi-free-start collisions for 8 rounds of the Grøstl-0-224 and -256 compressions functions using the Super-Sbox rebound

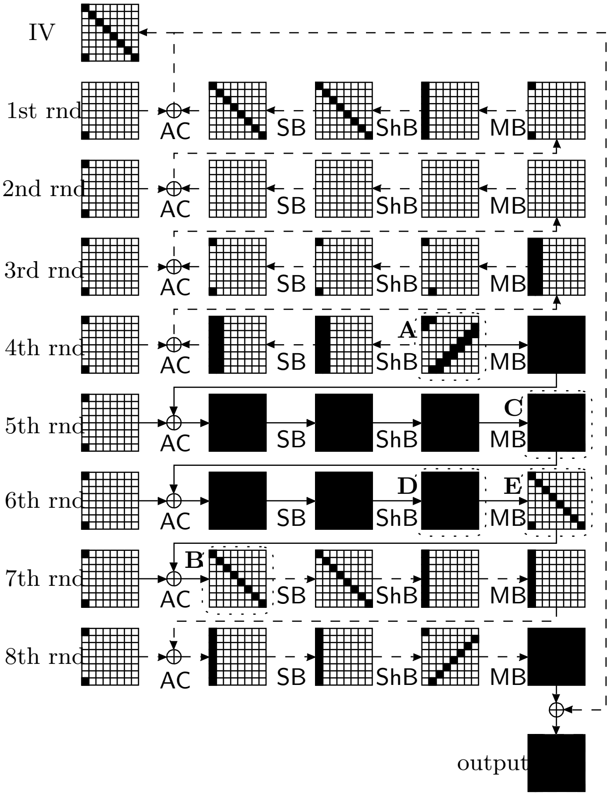


Figure 4.16: Differential path between P and Q of 8-round Grøstl-0-224 and -256

technique. The differential path between $P(m \oplus IV)$ and $Q(m)$ of the message block is shown in Figure 4.16.

We again denote the start of the controlled phase by A and the end by B. By the Super-Sbox technique, we can generate a solution for the controlled phase with complexity one on average.

For the probabilistic uncontrolled transitions, we observe that in the forward direction (from state B to the end), the path is followed with probability almost one. In the backward direction (from state A to the beginning), we need to follow two $8 \to 1$ transitions at the 3rd round inverse MB (accounting for 2^{112} computations), and 2^{16} time to follow the 3rd round inverse AC. The overall time complexity to find a message block following the whole path is hence 2^{128}.

In order to obtain such a pair with colliding CVs, we need to obtain a collision on the 8 active bytes before the final MB and the 8 active bytes in the IV. Consequently, about 2^{64} message blocks have to be generated to obtain a collision. This determines the total attack complexity as $2^{128+64} = 2^{192}$ for time and 2^{64} for memory.

By counting the degrees of freedom, we find that we can generate at most 2^{64} message blocks following the path, which is sufficient since our attack requires just 2^{64} message blocks following the path to obtain one collision of CV with good probability.

4.7.3 Attacks on Grøstl-0-512

In this section, we demonstrate collision attacks on the 512-bit hash length variant of the Grøstl-0 family.

Collisions for 6-round Grøstl-0-512

The internal differential path for our collision attack on 6 rounds of the Grøstl-0-512 hash function is shown in Figure 4.17.

Using the start-from-the-middle technique, the controlled rounds from state A to state B can be followed with average complexity one. The uncontrolled rounds require time $2^{16}, 2^{112}$ and 2^{16} to follow the transitions at the 2nd round inverse AC, the 1st round inverse MB, and the 1st round inverse AC, respectively. In the forward direction, the probability to follow the path is almost one.

For message blocks following the path, the output of the compression function depends only on the 10-byte difference at the input of the 6th round MB. The 2-byte difference among the 10 bytes comes from the 6th round AC. We observe that these 2-byte differences at the output of the 6th round AC are fixed, the number of possible differences of each byte at the output of the 6th round SB is only 2^7. Therefore, the output of the compression function lives in a $2 \cdot 7 + 8 \cdot 8 = 78$-bit vector space. In order to obtain a collision of CVs, we therefore need 2^{39} messages following the path. This implies that the total complexity of the attack is 2^{183} for time and 2^{64} for memory. By counting the degrees of freedom, we find that we can generate 2^{64} messages following the path, which is more than enough for the purposes of our attack.

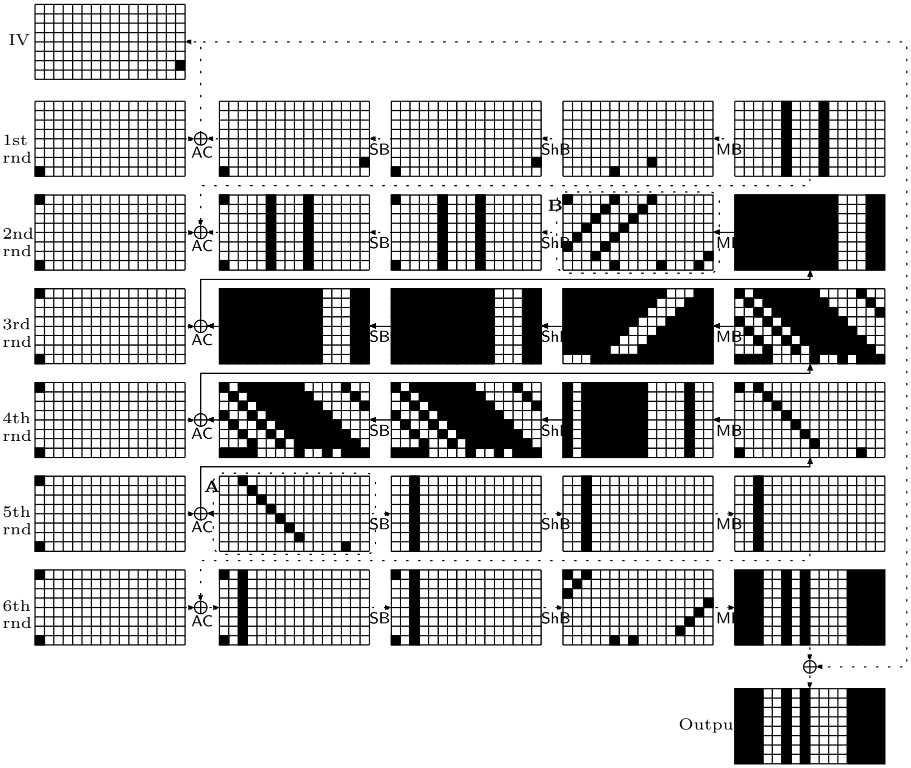


Figure 4.17: Differential path between P and Q of 6-round Grøstl-0-512

Semi-free start collisions for 8- and 9-round Grøstl-0-512

The path used for the collision attack on 6 rounds of the Grøstl-0-512 hash function can be extended to obtain semi-free-start collisions on the compression function reduced to 8 and 9 rounds. Both attacks use the Super-Sbox technique to span more active states in the middle.

The path which is used in the 8-round attack is depicted in Figure 4.18. For the uncontrolled phase, we need to follow the 3rd round inverse MB and inverse AC backwards (requiring 2^{112} and 2^{16} time, respectively), while the forward transitions occur with probability almost one. In order to obtain a collision of the chaining value, we need a collision on 24 bytes (8 from the difference in the IV and 16 from the active bytes before the last MB). Therefore, 2^{96} message blocks are required for a collision, accounting for a total attack complexity of $2^{128+96} = 2^{224}$ time and 2^{96} memory. Since the available degrees of freedom

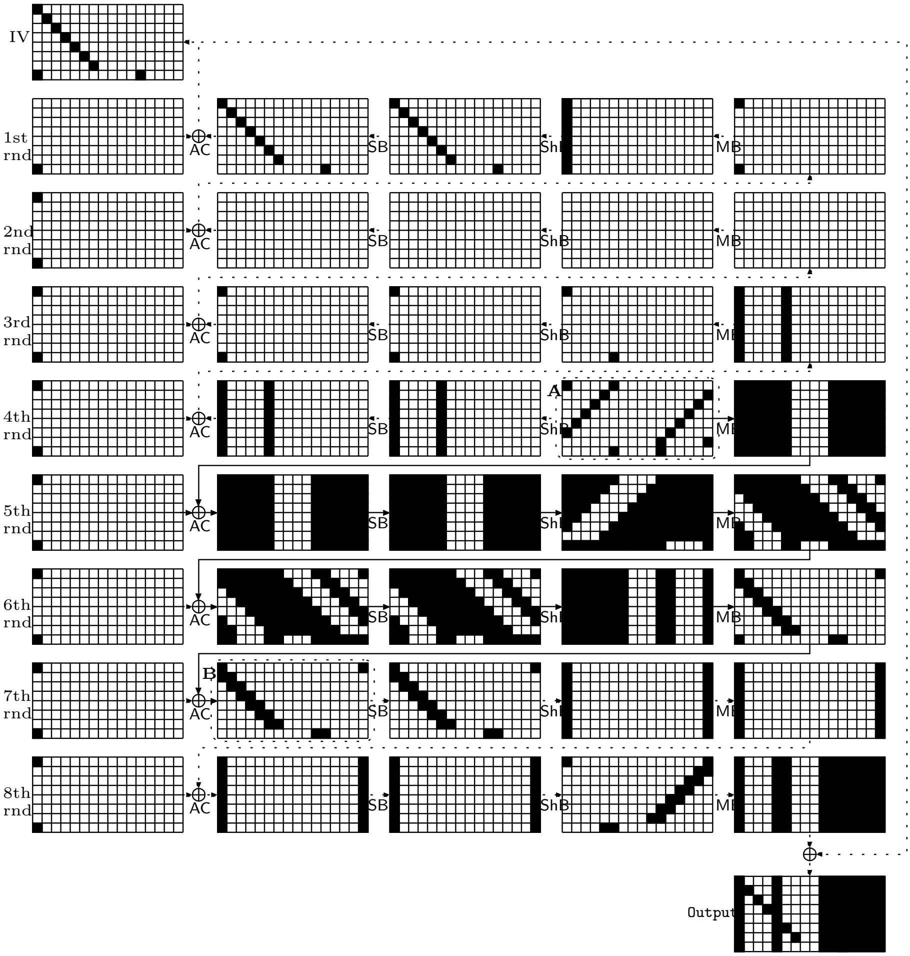


Figure 4.18: Differential path between P and Q of 8-round Grøstl-0-384 and -512

allow the generation of 2^{128} different message blocks following the path, our attack works.

By affording a fully active state in the middle and one more active column before the last MB, this attack can be extended to 9 rounds. The resulting path is shown in Figure 4.19. The uncontrolled phase has the same probability in the backward direction, accounting for a complexity of 2^{128}. In the forward direction, it takes 2^{240} time for the 7th-round MB and 2^8 time for the 8th-round AC. A message block following the path can therefore be generated with time complexity 2^{376}. The CV lives in a 256-bit vector space, implying that we need 2^{128} solutions for the path in order to obtain one collision. The available degrees of freedom allow for the construction of 2^{136} solutions, which is sufficient for our purposes. In total, this attack has a complexity of 2^{504} time and 2^{128} memory.

4.7.4 On memoryless collisions

Even though the collision attack on 5 rounds of Grøstl-0 has practical time complexity, the memory requirement of 2^{32} 512-bit blocks for finding two colliding message blocks is still substantial. A straightforward application of memoryless collision search seems to fail since the start-from-the-middle algorithm requires randomized iterations.

However, memoryless algorithms for the birthday problem such as Floyd's cycle finding algorithm [104] can be combined with the start-from-the-middle algorithm outlined in Sect. 3.1 as follows. We start with a randomly chosen 8-byte input difference in step 1. For the required repetitions of steps 1 and 2 before step 3 succeeds, we apply a "random" single-cycle bijection of $\mathbb{Z}/(2^{64}\mathbb{Z})$ to the previous differences, for instance $f(x) = 2x^2 + x + 1$. This is repeated until the first message block is obtained. Starting from the computation of the second message block, 8 diagonal bytes from the difference between P and Q from the previously obtained message block will be used as initial differences for step 1 instead of random values. Since those bytes have been processed through several SubBytes applications, it is reasonable to assume that they will behave as random data. In this way, the start-from-the-middle procedure itself acts as a deterministic mapping for the collision search. Under those assumptions, a colliding message block is expected to be found in a small multiple of 2^{32} iterations with constant memory.

In this case, the memory requirements of the collision attacks on 5 rounds and the semi-free-start collision attacks on 7 rounds of the Grøstl-0-224 and -256 hash functions become negligible without significant increase in time complexity.

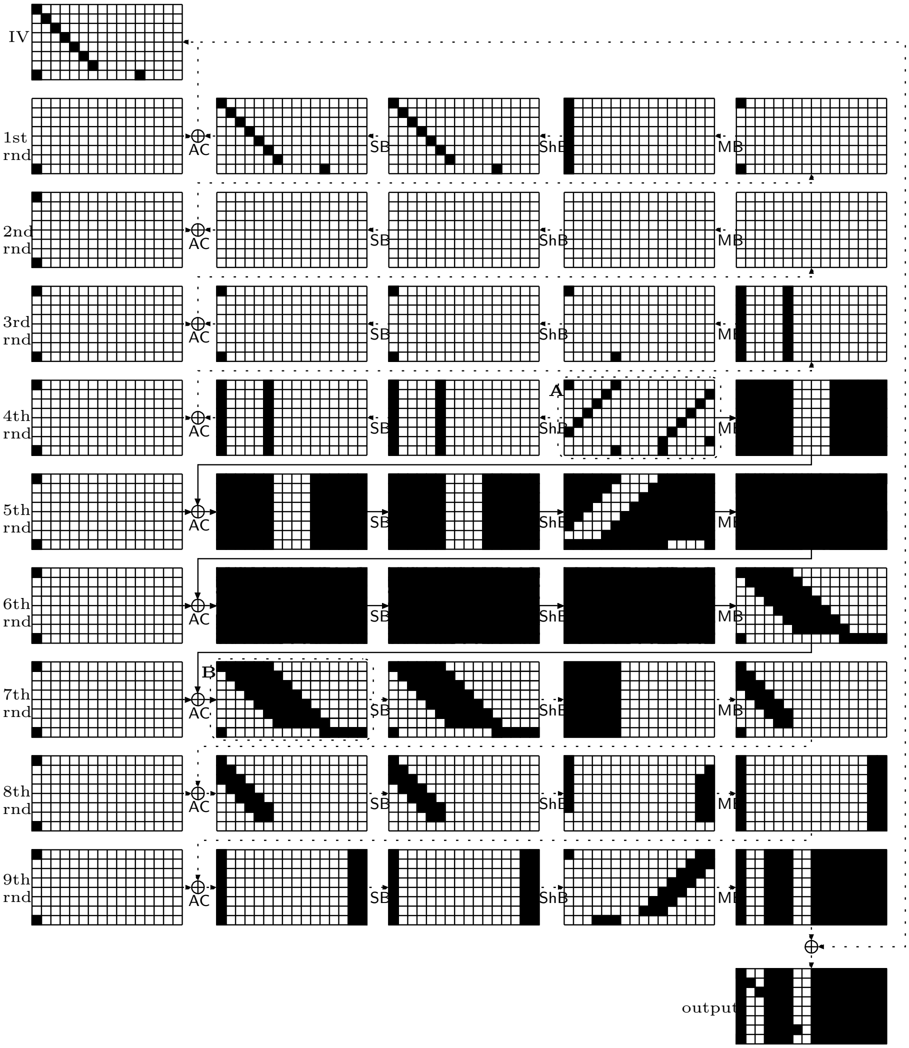


Figure 4.19: Differential path between P and Q of 9-round Grøstl-0-384 and -512

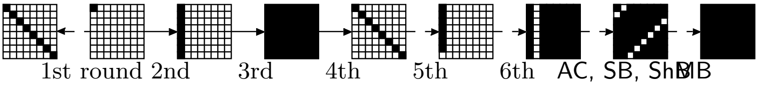


Figure 4.20: Differential path for the 7-round distinguisher of the Grøstl permutation.

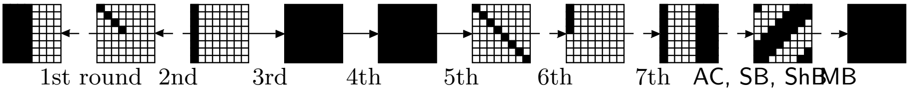


Figure 4.21: Differential path for the 8-round distinguisher of the Grøstl permutation.

Likewise, the memory requirements of the semi-free-start collision attacks on 8 rounds and 9 rounds of Grøstl-0-384 and -512 become 2^{64}. The memory requirements of the other attacks remain unchanged.

4.8 Distinguishers for the round-reduced Grøstl-0-256 permutation

In this section, we show improved distinguishers for 7 and 8 rounds of the Grøstl permutations, applicable to both P and Q. The distinguisher for seven rounds has practical time and memory requirements. Our distinguishers work in both the limited-birthday [77] and the stronger Subspace Problem model [106]. Before going into detail about the different notions of distinguishing, we describe the differential paths and attack procedures.

The truncated differential path for the Grøstl permutation reduced to 7 rounds is depicted in Figure 4.20. We use the start-from-the-middle technique to create a pair for this path with a complexity of about 2^8 compression function calls and negligible memory. Starting from the input to `SubBytes` in the 5th round, we can create a pair following the path back to the output of `SubBytes` in Round 2 with a complexity of about one. The remaining steps in both directions are uncontrolled. Except for the transition from 8 to 7 active bytes in the 5th round (which happens with probability 2^{-8}), they are followed with probability of almost one, hence about 2^8 repetitions are sufficient to generate one pair following the entire path.

For eight rounds, we use the truncated differential path illustrated in Figure 4.21. In order to afford the two fully active states in the middle, we

a	b
`1005d35dd6ae67e7`	`6905d35dd6ae67e7`
`be596c8193d8ca75`	`be536c8193d8ca75`
`cf8110ef1700dcab`	`cf8133ef1700dcab`
`be37b4d7581c2f2a`	`be37b49f581c2f2a`
`e5bd7afc7531c99d`	`e5bd7afc9831c99d`
`5073a23cf2065561`	`5073a23cf2b85561`
`341edf8988424101`	`341edf898842f501`
`d989073be1cb0af7`	`d989073be1cb0ace`

Table 4.4: Input pair (a, b) of Grøstl's P permutation following the trail used in the 7-round distinguisher.

employ the Super-Sbox technique. Starting from the output of `MixBytes` in round 3, 2^{64} pairs following the path until the input of `MixBytes` in round 6 can be generated at a cost of 2^{64} computations and 2^{64} memory. Out of these 2^{64} pairs, a fraction of about $2^{-2 \cdot 32}$ are expected to follow the required $8 \to 4$ transitions in both backward and forward direction. The remaining transitions have a probability of about one, so that one pair following the entire 8-round path can be found with 2^{64} time and memory.

4.8.1 Distinguishers in the limited-birthday model

A limited-birthday distinguisher for a keyed permutation consists of an efficient procedure to obtain pairs of inputs and outputs such that the input and output differences are zero at i and j bit positions, respectively. If this procedure is more efficient than the conceived best generic algorithm based on the birthday paradox, it is considered a valid distinguisher [77]. Although primarily targeted for a known-key setting for keyed permuations, limited-birthday distinguishers have been applied to the Grøstl permutation and compression function [77].

In this setting, obtaining input/output pairs for the seven-round Grøstl permutation (barring the last `MixBytes`) with a zero difference at 448 input and 64 output bits as depicted in Figure 4.20 should ideally take 2^{32} operations, while following our procedure has a complexity of 2^{8}. We have implemented the algorithm for the seven-round distinguisher. An example pair of inputs to the P permutation following the entire path can be found in Table 4.4.

Likewise, in the eight-round case (Figure 4.21), the ideal complexity is 2^{128}, while our procedure takes 2^{64} time and memory.

4.8.2 Distinguishers in the subspace problem model

Distinguishers based on the Subspace Problem [106] consider the problem of obtaining t difference pairs for an N-bit permutation such that the output differences span a vector space of dimension less than or equal to n (provided the input differences span a vector space of dimension less than or equal to n too). Contrary to the limited-birthday model, lower bounds for the number of permutation queries needed in the generic case can be proven [106], so this provides a stronger distinguishing setting. According to Corollary 4 of [106], the number of queries to the permutation and its inverse to solve the subspace problem is lower bounded by

$$Q \geq \begin{cases} \sqrt{2K} & \text{if } K < 2^{2n-1}, \\ 2^{-n} \cdot K & \text{if } K \geq 2^{2n-1} \end{cases} \tag{4.15}$$

with

$$K = \frac{t}{e} \left(p\sqrt{2\pi t} \right)^{1/t} \cdot 2^{\frac{(N-n)(t-2n)-2(n+1)}{t}}. \tag{4.16}$$

Note that the conditions on t and N stated in Proposition 2 of [106] can actually be relaxed to $t > 2n$ and $N \geq n$.

For the Grøstl permutation, we have $N = 512$. Our procedure to generate pairs for the seven round trail of Figure 4.20 has to be compared to the generic lower bound given by (4.15) with $n = 448$. Starting from $t = 2^{11}$, our method is more efficient ($2^{8+11} = 2^{19}$ computations) than the generic algorithm (2^{23} queries), yielding a valid distinguisher.

For eight rounds, we have $n = 256$, and again choosing $t = 2^{11}$ gives a complexity of 2^{101} for the generic case, while our method has a complexity of $2^{64+11} = 2^{75}$ time and 2^{64} memory.

4.9 Conclusions

In this chapter, we studied hash functions based on the wide trail strategy with a special focus on their resistance to the powerful class of rebound attacks.

First, we described the analysis of Wildpool, a design study intended to evaluate the consequences of introducing bigger S-boxes in wide-trail designs. We observed that the interplay between the S-box and the diffusion layer restricted the usefulness of the construction and also obstructed its analyzability.

Furthermore, Wildpool succumbs to the rebound attack as much as its role model, Whirlpool.

Second, we presented the Whirlwind hash function design which takes into account recent developments in hash function cryptanalysis, in particular the rebound attack. It uses larger 16-bit S-boxes for low-probability differential trails. An implementation can still be efficient due to a special choice of basis for the finite field GF(2^{16}) speeding up implementations that compute the S-box entries instead of storing them in a large lookup table.

Third, we analyzed the Grøstl-0 hash function family, which is the original untweaked submission to the NIST SHA-3 competition. We presented improved collision attacks for 6 out of 10 rounds of Grøstl-0-224 and Grøstl-0-256 and 6 out of 14 rounds of Grøstl-0-512. We also identify weaknesses in reduced round versions of the compression function and the permutation. In part also to preclude our attacks, the Grøstl hash function was tweaked for the final rounds. Consequently, our results do not threaten the version of Grøstl contending in the SHA-3 final. More precisely, the tweaked version of Grøstl exhibits bigger differences between the permutations P and Q than Grøstl-0. These tweaks seem to make our attack strategy a lot less powerful.

Chapter 5

Block ciphers and statistical cryptanalysis

Statistical cryptanalysis techniques, most prominently including linear and differential cryptanalysis as described in Section 2.3, are among the most powerful and versatile analysis methods for symmetric algorithms. This particularly holds for block ciphers, not the least because these attacks were initially developed for the cryptanalysis of block ciphers [22,115,116], the most well-understood symmetric primitive at the time of their invention.

Since their discovery, they have received a lot of attention from the research community. Many successful [19,21,109] and even practically implementable attacks [23,26,116] on block ciphers are based on those two strategies and their refinements.

Also from a theoretical point of view, significant advances have been made in their understanding. The current state-of-the-art allows cryptographers to design ciphers that can be shown to resist at least the basic forms of these attacks [53,56,101,117,135,144,159].

However, despite these advances, our theoretical understanding of both linear and differential cryptanalysis still leaves a lot to be desired. For instance, the success probabilities and data complexities of linear and differential attacks are usually determined in a model based on several important assumptions to simplify the analysis [27,29,30,152,153], with a greatly varying level to which these assumptions and simplifications are backed up by supporting theoretical or experimental evidence [30, 152].

Additionally, little is known about precise success probability and data complexity estimates for extensions of basic differential and linear cryptanalysis such as differential-linear [107], truncated differential [102] and impossible differential cryptanalysis [19, 103], linear cryptanalysis with multiple approximations [24, 116] and multidimensional linear cryptanalysis [85]. Only recently, Blondeau, Gérard and Tillich provided the first theoretically founded estimates for the data complexity of differential-linear, truncated and impossible differential cryptanalysis [30]. However, these estimates are entirely asymptotic in nature and usually do not provide the necessary level of precision for the evaluation of a concrete attack with parameters used in the real world.

In order to remedy some of the limitations of previous studies, we note that while the correlation of linear approximations and the cardinality of differentials — the nonlinearity measures on which differential and linear attacks are based (see Sections 2.3.1 and 2.3.2) – are very well studied in the context of Boolean functions, it was only recently that their full distributions over the set of all Boolean functions or permutations of an arbitrary but fixed dimension were systematically studied by Daemen and Rijmen [58, 60], based on important earlier work by O'Connor [136–138].

Since both linear and differential cryptanalysis are statistical in nature in the sense that they use these nonlinearity measures — correlation and differential cardinality, respectively — to distinguish the concrete symmetric-key algorithm at hand from a randomly chosen Boolean function or permutation, the knowledge of these distributions is crucial for evaluating linear and differential attacks. In a word, they define the "ideal" behaviour that the designer is attempting to achieve, and deviations from which can be only exploited by the attacker if they are statistically significant enough to yield a valid distinguisher from the random case. However, to the best of our knowledge, the consequences of taking these full probability distributions into account when analysing statistical attacks on block ciphers, have not been studied so far. Previous analyses [27, 29, 30, 152, 153] have always assumed some notion of "average behaviour" instead of dealing with the full distributions. Similarly, the fact that correlation and differential probability typically vary over the key space, has often been neglected.

As we demonstrate in this chapter, the impact of the knowledge of these distributions in statistical cryptanalysis is vast. The complexity of a statistical attack exploiting a certain deviation from the ideal "random" behaviour intimately depends on them, a fact that was previously not taken into account. By working with these distributions instead of with simplifying assumptions, we are able to obtain a more accurate and deeper statement about how to evaluate linear attacks on block ciphers (Section 5.2).

At the same time, they enable us to provide a comprehensive model for the evaluation of so-called *structure attacks*, a sophisticated variant of differential cryptanalysis. Notably, this model, developed in Section 5.6, takes the varying differential probabilities over the keys into account.

5.1 Analysing the complexity of linear attacks

5.1.1 Problem statement

When speaking of linear cryptanalysis, we refer to Matsui's "Algorithm 2" [115], as outlined in Section 2.3.2.

Consider a linear attack on an iterative block cipher with a block length of n bits and R rounds, using a linear approximation (α, β) with bias $\epsilon \neq 0$ covering $r < R$ inner rounds (more specifically, rounds $(i, \ldots, i+r)$ for some $0 \leq i \leq R-r$). Suppose the attacker has obtained a number N of plaintext/ciphertext pairs encrypted under the same cipher key K (also called *samples*), and that the attacker wants to recover a subset of m key bits of the round subkeys involved in the rounds not covered by the linear approximation. By partially de- and/or encrypting the ciphertext and/or plaintext from each pair with each possible value of the m subkey bits, the approximation (α, β) is evaluated, and a counter T_i is maintained for each key candidate k_i, $0 \leq i < 2^m$. After this step, the key candidates are ranked in increasing order of the absolute value of the sample bias $|\widehat{\epsilon}_i| \stackrel{\text{def}}{=} |T_i/N - 1/2|$. Let $\widehat{\zeta}_i$ denote the $|\widehat{\epsilon}_i|$ sorted in increasing order.

Following the terminology of [152], we consider the attack successful if the correct key k_r is ranked among the highest 2^l out of the 2^m key candidates with some probability P_S:

Definition 5.1. A linear attack using Matsui's Algorithm 2 recovering m subkey bits provides an *advantage* of

$$a \stackrel{\text{def}}{=} m - l \tag{5.1}$$

bits over exhaustive search if the absolute sample bias $\widehat{\zeta}_r$ corresponding to the counter T_r associated with the correct key k_r is among the 2^l largest of the 2^m counters $T_i, 0 \leq i < 2^m$. The probability of this event:

$$P_S \stackrel{\text{def}}{=} \Pr\left(\widehat{\epsilon}_r > \widehat{\zeta}_{2^m - 2^l}\right), \tag{5.2}$$

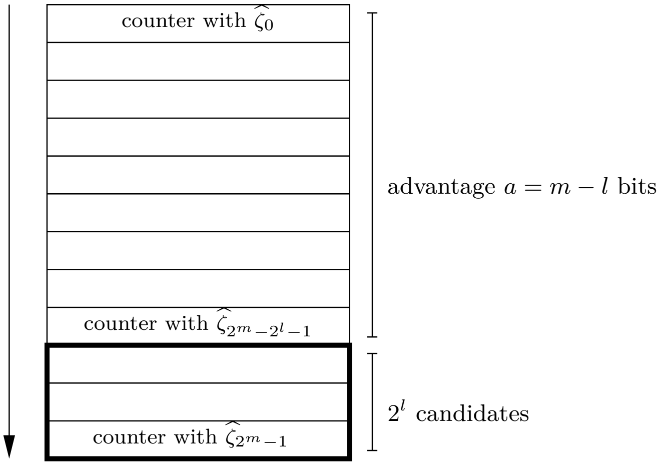

counter with $\widehat{\zeta_0}$
counter with $\widehat{\zeta_{2^m-2^l-1}}$
counter with $\widehat{\zeta_{2^m-1}}$
advantage $a = m - l$ bits
2^l candidates

Figure 5.1: Advantage of a linear attack

is called the *success probability* of the attack.

Note that the advantage a and the success probability P_S are tightly related by equations (5.1) and (5.2): An attack provides a certain advantage with a certain success probability; the same attack on the same number of subkey bits can (and typically will) have a different success probability when a different level of advantage over exhaustive search is sought. This terminology is illustrated in Figure 5.1.

Due to the stochastic nature of linear cryptanalysis, the counters T_i can be viewed as discrete random variables taking on values between 0 and N, the number of samples processed for each key guess k. Let $E_j[k]$, $1 \leq j \leq N$, denote the binary random variable taking on the value one if and only if the linear approximation (α, β) used in the attack holds for the j-th of the N plaintext/ciphertext pairs with key guess k. The counter T_i corresponding to key guess i is then given by

$$T_i = \sum_{j=1}^{N} E_j[i]. \tag{5.3}$$

The execution of Matsui's Algorithm 2 can now be viewed as a method of distinguishing the distribution of the T_i, $i \neq r$, arising for each of the $2^m - 1$ wrong key guesses from the distribution of the right key counter T_r. In this context, Junod and Vaudenay [98] have proved that the key ranking procedure

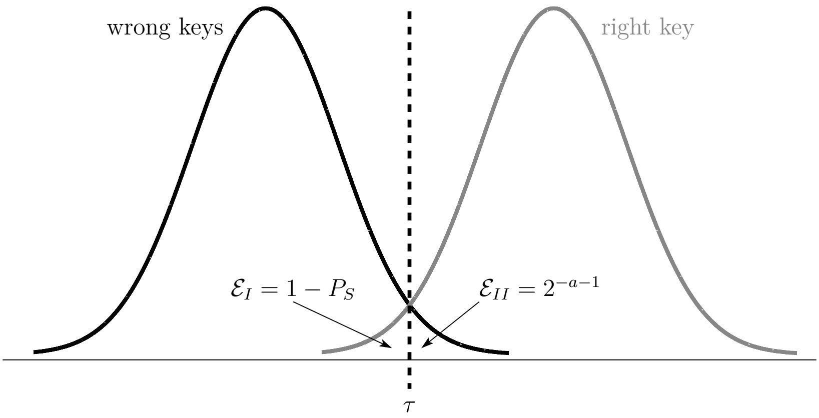


Figure 5.2: Relation between the type I and II error probabilities, success probability P_S and advantage a.

provided by Matsui's Algorithm 2 is optimal in the case of standard linear cryptanalysis.

Fixing the advantage a corresponds to a certain threshold τ such that key candidates i with counters $T_i \geq \tau$ will be included in the list of 2^l key candidates. As in standard hypothesis testing, two kinds of errors are possible:

- **Type I error:** This error occurs if the correct key k_r remains undetected. We denote the type I error probability by $\mathcal{E}_I$.
- **Type II error:** Symmetrically, this error occurs if a wrong key is suggested to be correct. We denote the type II error probability by $\mathcal{E}_{II}$.

These error probabilities can directly be related to the terminology of advantage and success probability: The success probability P_S is given by $P_S = 1 - \mathcal{E}_I$, and the advantage a is related to the type II error probability by $\mathcal{E}_{II} = 2^{-a-1}$. This also confirms the intuition that, for instance, an advantage of $a = 1$ bits corresponds to an attack suggesting a wrong key with a probability of $2^{-1-1} = 1/4$, which is a factor of $(1/2)^a = 1/2$ ("one bit") better than random guessing.

The type I and type II error probabilities, and correspondingly a and P_S, can be interpreted graphically as the overlapping areas between the probability density functions of the T_i for a wrong key k_i and the right key k_r. This is depicted in Figure 5.2.

5.1.2 Common assumptions

In order to obtain a meaningful statement about the distribution of the counters T_i for both the right key and the wrong keys, the general formulation of the counters as sum of binary random variables (5.3) has to be instantiated with concrete probability distributions. Additionally, further assumptions might be required to simplify the analysis.

First of all, the ranking of the key candidates by absolute sample bias depends on what is known as the *wrong-key randomisation hypothesis* [82, 96]:

Definition 5.2 (Wrong key randomisation hypothesis)**.** Consider a nontrivial linear approximation $\mathcal{L} = (\alpha, \beta)$ with absolute bias $|\epsilon| \gg 0$ for virtually all possible cipher keys. Let k_r be the right subkey guess. Then, for virtually all cipher keys and for all wrong subkey guesses $k_w \neq k_r$:

$$\frac{\left|\Pr(\mathcal{L} \text{ holds} \mid k_r) - \frac{1}{2}\right|}{\left|\Pr(\mathcal{L} \text{ holds} \mid k_w) - \frac{1}{2}\right|} \gg 1.$$

This assumption basically requires the absolute sample bias for all wrong key guesses to be lower than for the right key guess. The intuition behind it is that partially de- and/or encrypting a small number of rounds with a wrong key is resulting in essentially random behaviour, a common interpretation being that the linear approximation is going to hold with a bias of zero for all wrong keys [30, 115, 152]. We call this stronger version of Definition 5.2 the *standard wrong key randomisation hypothesis*:

Definition 5.3 (Standard wrong key randomisation hypothesis)**.** Consider a nontrivial linear approximation $\mathcal{L} = (\alpha, \beta)$ with absolute bias $|\epsilon| \gg 0$ for virtually all possible cipher keys. Let k_r be the right subkey guess. Then, for virtually all cipher keys and for all wrong subkey guesses $k_w \neq k_r$:

$$\left|\Pr(\mathcal{L} \text{ holds} \mid k_w) - \frac{1}{2}\right| \approx \frac{1}{2}.$$

A second consequence of the wrong-key randomisation hypothesis is that it is commonly assumed [30] that for all wrong keys k_w and all plaintext/ciphertext samples j, the binary random variables $E_j[k_w]$ follow the same Bernoulli probability distribution $\mathrm{Bern}(p_0)$ (with $p_0 = 1/2$ being the common assumption). Since the individual plaintext/ciphertexts pairs are assumed to be statistically independent, this results in a binomial distribution $\mathrm{Bin}(N, p_0)$ for the counters T_{k_w} for all wrong key guesses k_w. For $N \geq 5$ and p_0 close to $1/2$, this can be tightly approximated by a normal distribution $\mathcal{N}(Np_0, Np_0(1 - p_0))$ [154].

Dually to the wrong key randomisation hypothesis, a common assumption for the statistical behaviour when partially de- and/or encrypting with the right key is that the bias of the resulting linear approximation does not deviate significantly from its average over all keys [82,96]:

Definition 5.4 (Key equivalence hypothesis)**.** Consider a nontrivial linear approximation $\mathcal{L} = (\alpha, \beta)$ with absolute bias $|\epsilon| \gg 0$ for virtually all of the $|\mathcal{K}|$ possible cipher keys. Then, the key-dependent bias of $\mathcal{L}$ is virtually independent of the choice of the key:

$$\left|\Pr(\mathcal{L} \text{ holds} \mid k_0) - \frac{1}{2}\right| \approx 2^{-|\mathcal{K}|} \sum_{k \in \mathcal{K}} \left|\Pr(\mathcal{L} \text{ holds} \mid k) - \frac{1}{2}\right| \qquad \forall k_0.$$

5.1.3 Previous work

The fact that many published linear attacks have data and time requirements beyond practical reach (see for instance [139] and [109]) implies that the question of how to accurately estimate their complexity — and hence determine which attack actually is a valid attack — is of great importance to the security of block ciphers.

As a result, the data complexity of both Matsui's Algorithm 2 and the more general problem of distinguishing the distributions for the right and the wrong keys have been extensively studied in the literature [9,10,30,96–98,115,116,152].

In his original papers, using a normal approximation to the binomial distribution, Matsui [115,116] estimates the data complexity to be of the order $\frac{1}{4}\epsilon^{-2}$ and gives estimations which multiple of this is required to obtain a certain success probability. This analysis has been systematised and deepened by Junod [96]. Furthermore, Junod and Vaudenay [98] have proven that Matsui's key ranking procedure is optimal for the case of Algorithm 2 using a single linear approximation.

In his important work, Selçuk [152] presented a thorough statistical analysis of the data complexity of linear and differential attacks based on a model of Junod [96] and a normal approximation for order statistics. This yields practical closed formulas for the success probability P_S and data complexity N of a linear attack when an advantage of a bits is sought:

Theorem 5.5 ([152, Theorem 2])**.** *Let P_S be the probability that a linear attack on an m-bit subkey, with a linear approximation of probability p, with N known plaintext blocks, delivers an a-bit or higher advantage. Assuming that the linear approximation's probability to hold is independent for each key tried and is equal* *to* $1/2$ *for all wrong keys, one has for sufficiently large* m *and* N*:*

$$P_S = \Phi\left(2\sqrt{N}|p-1/2| - \Phi^{-1}(1-2^{-a-1})\right). \tag{5.4}$$

Corollary 5.6 ([152, Corollary 1])**.** *With the assumptions of Theorem 5.5,*

$$N = \left((\Phi^{-1}(P_S) + \Phi^{-1}(1-2^{-a-1}))/2\right)^2 \cdot |p-1/2|^{-2} \tag{5.5}$$

plaintext blocks are needed in a linear attack to accomplish an a-bit advantage with a success probability of P_S*.*

Generalizing linear attacks based on Matsui's Algorithm 2 and other statistical cryptanalyses, Junod [97], Baignères, Junod and Vaudenay [9] and Baignères and Vaudenay [10] have derived asymptotic formulas for the required data complexity of optimal distinguishers between distributions. In this setting, the error probabilities of the statistical test are related to our terminology as follows. As noted in Section 5.1.1, the type I error is equal to $1 - P_S$, while the type II error probability is given by 2^{-a-1}. Theorem 6 of [9] essentially yields the following estimate for the data complexity:

$$N \approx \frac{2\Phi^{-1}\left(\frac{(1-P_S)+2^{-a-1}}{2}\right)^2}{D(p||0.5)}, \tag{5.6}$$

with p denoting the probability of the linear approximation and $D(p||0.5)$ denoting the Kullback-Leibler divergence [30] between two binomial distributions with respective probabilities p and 0.5. Corollary 4 of [10] essentially gives the following asymptotic estimate for N:

$$N \approx -\frac{\ln \max\{1-P_S, 2^{-a-1}\}}{D(p||0.5)}. \tag{5.7}$$

Recently, Blondeau, Gérard and Tillich [30] have developed a unified framework for estimating the data complexity of various statistical attacks. This framework is based on a more precise estimation of binomial tails, which is especially relevant in the case of differential cryptanalysis [30, 152]. (In the case of linear cryptanalysis, the normal approximation is very precise already for small N.) For the evaluation of N, a numerical procedure called "Algorithm 1", and first and second order estimates N' and N'' for $P_S = 1/2$ are provided in [30]. We restate the estimate N' for the case of linear cryptanalysis [30, Theorem 2]:

$$N' = -\frac{1}{D(p||0.5)}\left(\ln\left(\frac{\nu \cdot 2^{-a-1}}{\sqrt{D(p||0.5)}}\right) + 0.5\ln\left(-\ln\left(\nu \cdot 2^{-a-1}\right)\right)\right), \tag{5.8}$$

where $\nu = \Big((p - 0.5)\sqrt{2\pi(1-p)}\Big) \,/\, \big(\sqrt{p}/2\big)$.

Note that throughout the literature, the assumption is made that decrypting with a wrong key results in a zero bias for the linear approximation. As we will see, this constitutes a simplified view of the problem.

5.1.4 Distribution of biases in Boolean permutations

Daemen and Rijmen [60] have proved the following characterisation of the distribution of correlation of a fixed linear approximation over the set of all n-bit permutations:

Theorem 5.7 ([60, Theorem 4.7])**.** *Consider a fixed nontrivial linear approximation (α, β) with $\alpha, \beta \neq 0$. When $n \geq 5$, the distribution of the correlation $C_{\alpha,\beta}$ over all n-bit permutations can be approximated by the following distribution up to continuity correction:*

$$C_{\alpha,\beta} \sim \mathcal{N}(0, 2^{-n}). \tag{5.9}$$

Since $C = 2\epsilon$, this immediately implies

Corollary 5.8. *With the assumptions of Theorem 5.7,*

$$\epsilon_{\alpha,\beta} \sim \mathcal{N}(0, 2^{-n-2}). \tag{5.10}$$

We therefore note that if we fix an arbitrary nontrivial linear approximation, its bias will be normally distributed over all n-bit Boolean permutations with mean 0 and variance 2^{-n-2}.

5.1.5 Motivation for a re-evaluation

Being the basis of all previous analyses of linear attacks using Matsui's Algorithm 2, the assumptions listed in Section 5.1.2 can be considered standard practice. Even more importantly, they seem to provide an adequate basis for practical evaluations of the success probability and data complexity of concrete linear attacks. Especially the framework developed by Selçuk [152], yielding convenient closed formulae, is very suitable for this purpose, with the additional benefit of being supported by experimental evidence on small-scale linear attacks.

The intuition behind the common form of the wrong key randomisation hypothesis (Definition 5.3) is that partial de- and encryption with the wrong key yields the evaluation of the linear approximation on a randomly drawn permutation. It is then assumed that this will result in a zero bias most of the time. While Corollary 5.8 confirms that zero is the most likely single value of the bias to occur, this is merely the expected value of a Gaussian distribution. Previous analysis has therefore replaced the full distribution by its mean. While some form of wrong key randomisation assumption seems inevitable, it remains open to what extent this simplification is justified. We will study this question in detail in Sections 5.2 and 5.3.

Additionally, the experiments conforming the adequacy of the wrong key randomisation hypothesis in its simplest (and coincidentally, also most extreme) form all deal with linear approximations with very high bias compared to the block size n. With a data complexity of roughly $\frac{1}{4}\epsilon^{-2}$, the minimum exploitable bias would be of the order $2^{-n/2-1}$ before the attacker exceeds the available plaintext space. In practice, attacks are usually pushed to cover the maximum number of rounds of a cipher, implying a bias that is much closer to this limit than in the experiments. We can therefore conclude that, somewhat surprisingly, the implications of the standard interpretation of the wrong key randomisation hypothesis have not been tested in the range of parameters that is of the greatest practical interest.

Compared to the standard wrong key randomisation assumption, the validity of the key equivalence hypothesis is clearly more contested. Since the discovery of the linear hull effect by Nyberg [133, 134], it is known that the bias can heavily depend on the key. The impact of this has recently been confirmed by Leander [109]. As a result, it becomes virtually infeasible to predict the exact bias for the right key for block ciphers with real-world parameters. At the same time, however, every framework for the analysis of linear attacks needs to assume that this bias can be computed reliably. In Section 5.4, we will study a heuristic that aims at making the complexity estimation in linear cryptanalysis more realistic by taking the linear hull effect into account.

5.2 An improved model for the analysis of linear attacks

5.2.1 More accurate wrong key randomisation

For the analysis of linear attacks using Matsui's Algorithm 2, we use the notation introduced in Section 5.1.1: In an attack on an n-bit block cipher using a linear approximation (α, β) using N known plaintexts, a counter T_i is maintained for each key candidate k_i.

The distribution of T_i has a critical impact on the precision of estimating the data complexity of Matsui's Algorithm 2. First of all, the distribution of the T_i for the wrong keys has to be determined.

Standard wrong key randomisation hypothesis

To the best of our knowledge, all previous complexity evaluations of Matsui's Algorithm 2 are based on the wrong key randomisation hypothesis in its form of Definition 5.3, assuming zero bias for all wrong keys. Note that since this minimises overlap between the distributions for the wrong and for the right keys, this constitutes the best-case scenario from the attacker's point of view.

In this case, making the usual independence assumption, the distribution of the wrong key counters T_w is given by a binomial distribution with probability $p = 1/2$ and N repetitions. For sufficiently large N, this can be very closely approximated by a normal distribution with mean $Np = N/2$ and variance $Np(1-p) = N/4$. The sample bias $\widehat{\epsilon}_w = T_w/N - 1/2$ of the wrong keys is therefore assumed to be approximately distributed as $\mathcal{N}(0, 1/(4N))$.

Adjusted wrong key randomisation hypothesis

Though the standard formulation of the wrong key randomisation hypothesis is inspired by the intention to make the approximation (α, β) behave as for a randomly drawn n-bit permutation, the distribution of the $\widehat{\epsilon}_w$ is not completely adequate. In fact, it is known (see Theorem 5.7 and Corollary 5.8) that the bias of (α, β) over the n-bit permutations is not constantly zero, but instead follows a known distribution over the wrong keys. The following lemma, which is a new result, takes this into account.

Lemma 5.9. *In a linear attack with Matsui's Algorithm 2 on an n-bit block cipher using N known plaintexts, the sample bias $\widehat{\epsilon}_w$ of the wrong keys approximately follows a normal distribution with mean zero and variance $1/4 \cdot (1/N + 1/2^n)$:*

$$\widehat{\epsilon}_w \sim \mathcal{N}(0, 1/4\left(\frac{1}{N} + \frac{1}{2^n}\right)). \tag{5.11}$$

Proof. For the decryption with a wrong key k_w, the linear approximation is expected to behave as in the case of a randomly drawn n-bit permutation. According to Corollary 5.8, for each of the N samples, we therefore obtain a bias ϵ_w drawn from the normal distribution $\mathcal{N}(0, 2^{-n-2})$. For the bias over N

samples, we therefore have

$$\widehat{\epsilon}_w \sim \mathcal{N}(\epsilon_w, \frac{1}{4N})$$

with $\epsilon_w \sim \mathcal{N}(0, 2^{-n-2})$. Consequently,

$$\widehat{\epsilon}_w \sim \mathcal{N}(0, 2^{-n-2}) + \mathcal{N}(0, \frac{1}{4N}) \quad = \mathcal{N}(0, 1/4\left(\frac{1}{N} + \frac{1}{2^n}\right)),$$

as claimed. □

The impact of the new wrong key randomisation hypothesis is illustrated in Figure 5.3.

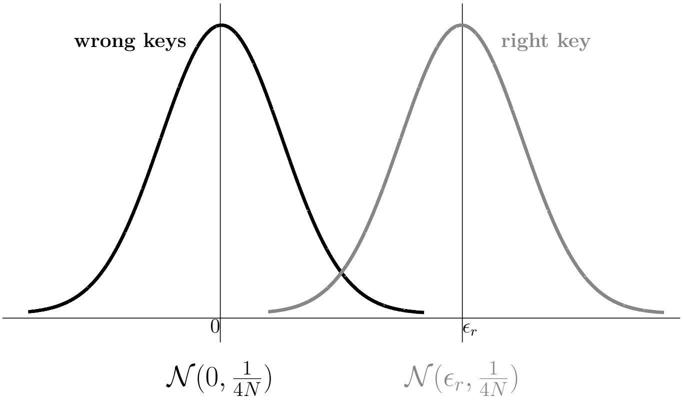
Previous interpretations of the wrong key randomisation hypothesis have therefore used the mean zero instead of the full distribution $\mathcal{N}(0, 2^{-n-2})$ for the bias when decrypting with a wrong key. For the sample bias of the wrong keys, this resulted in using $\mathcal{N}(0, 1/(4N))$ instead of $\mathcal{N}(0, 1/4\left(\frac{1}{N} + \frac{1}{2^n}\right))$, implying that the distributions for the right key and the wrong keys were assumed to only differ in the mean, but had the same variance. While this arguably simplifies the analysis, the possible impact of this simplification has to be investigated.

Experimental verification

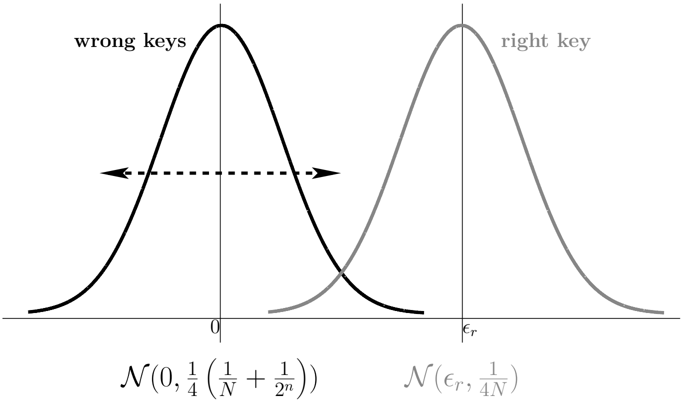
Even in the new form presented in Lemma 5.9, the wrong key randomisation hypothesis remains an idealisation. In order to verify that it reflects the reality with reasonable accuracy, we have experimentally determined the distribution of the sample bias over 2^{16} wrong keys for two structurally very different small-scale ciphers with a block length of 20 bits: SMALLPRESENT-20 [108] with 8 rounds, and RC6-5/6/10 with four 5-bit words, 6 rounds and an 80-bit key. In both cases, the number of samples was $N = 2^{16}$. As illustrated in Figure 5.4 the resulting distributions follow the theoretical estimate of (5.11) quite closely in both cases. Note that the scattering of data points occurs due to the fact that we are basically using a histogram with bin size one, and deal with raw data instead of averaging.

5.2.2 Probability of success

In this section, we study the implications of Lemma 5.9 for the success probability in linear cryptanalysis with Matsui's Algorithm 2. This leads to a new formula for the success probability of a linear attack.

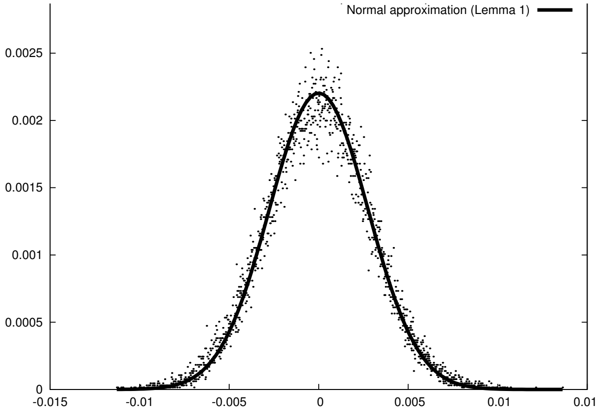


(a) Standard WKRH

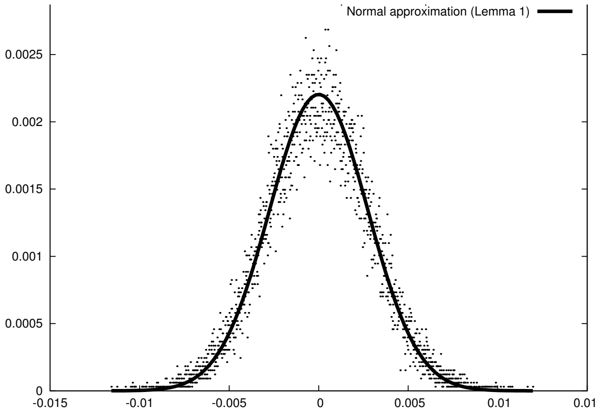


(b) Adjusted WKRH

Figure 5.3: Comparison of the standard and new wrong key randomisation hypotheses (WKRH).



(a) SmallPresent-20, 8 rounds



(b) RC6-5/6/10

Figure 5.4: Experimental distribution of the sample bias over 2^{16} wrong keys and 2^{16} texts for SmallPresent and small-scale RC6.

Theorem 5.10. *Consider a linear attack with Matsui's Algorithm 2 on an n-bit block cipher ($n \geq 5$) using a linear approximation with bias $\epsilon \neq 0$ and sufficiently large $N \leq 2^n$ known plaintexts. Denote by P_S the probability that this attack succeeds with an advantage of $a > 0$ bits over exhaustive key search. Then*

$$P_S \approx \Phi\left(2\sqrt{N}|\epsilon| - \sqrt{1 + \frac{N}{2^n}}\Phi^{-1}(1 - 2^{-a-1})\right). \tag{5.12}$$

Proof. We follow the approach of [152] and recall some notation from Sections 2.3.2 and 5.1.1. The advantage $a = m - l$, with m the number of key bits recovered by the attack, and l the length of the list of the top-ranking key candidates kept by the adversary. Let T_i, $0 \leq i < 2^m$, denote the counter associated with the key candidate k_i. Let T_r be the counter of the right key k_r, and denote an arbitrary wrong key counter by T_w. The attack then ranks the keys based on the absolute sample biases of the key candidates $|\widehat{\epsilon}_i| = |T_i/N - 1/2|$. Let $\widehat{\zeta}_i$ denote the $|\widehat{\epsilon}_i|$ sorted in increasing order. Define $L \overset{\text{def}}{=} 2^m - 2^l = 2^m - 2^{m-a}$. An a-bit advantage is now achieved if $\widehat{\epsilon}_r > \widehat{\zeta}_L = \widehat{\zeta}_{2^m - 2^l}$, that is, if the sample bias of the right key is large enough to be in the list of 2^l highest ranking candidates. In other words,

$$P_S = \Pr\left(\widehat{\epsilon}_r - \widehat{\zeta}_L > 0\right). \tag{5.13}$$

The sample bias of the right key approximately follows a normal distribution $\mathcal{N}(\mu_r, \sigma_r^2)$ with $\mu_r = \epsilon$ and $\sigma_r^2 = 1/(4N)$. According to Lemma 5.9, the absolute values of the sample biases for the wrong keys follow a folded normal distribution with mean $\mu_w = 0$ and variance $\sigma_w^2 = 1/4 \cdot (1/N + 2^{-n})$. Theorem 1 of [152] now yields that $\widehat{\zeta}_L = \widehat{\zeta}_{2^m - 2^l}$ is approximately normal distributed with

$$\mu_L = \mu_w + \sigma_w \Phi^{-1}(1 - 2^{-a-1})$$

and

$$\sigma_L^2 = \frac{\sigma_w^2}{\left(2\phi(\Phi^{-1}(1 - 2^{-a-1}))\right)^2} \cdot 2^{-m-a}.$$

Consequently, the random variable $\left(\widehat{\epsilon}_r - \widehat{\zeta}_L\right)$ follows a normal distribution $\mathcal{N}(\mu_r - \mu_L, \sigma_r^2 + \sigma_L^2)$. As noted in [152], the influence of σ_L^2 is negligible, and we can therefore express the success probability (5.13) in terms of a standard normal distribution as follows:

$$
\begin{aligned}
P_S &= \Pr\left(\widehat{\epsilon}_r - \widehat{\zeta}_L > 0\right) \\
&\approx \int_{-\frac{\mu_r-\mu_L}{\sqrt{\sigma_r^2+\sigma_L^2}}}^{\infty} \phi(x)dx \\
&\approx \int_{-\frac{\mu_r-\mu_L}{\sigma_r}}^{\infty} \phi(x)dx \\
&= 1 - \int_{-\infty}^{-\frac{\mu_r-\mu_L}{\sigma_r}} \phi(x)dx \\
&= \Phi(\frac{\mu_r-\mu_L}{\sigma_r}) \\
&= \Phi\left(\frac{|\epsilon| - 1/2\sqrt{\frac{1}{N}+\frac{1}{2^n}}\Phi^{-1}(1-2^{-a-1})}{1/(2\sqrt{N})}\right) \\
&= \Phi\left(2\sqrt{N}|\epsilon| - \sqrt{1+\frac{N}{2^n}}\Phi^{-1}(1-2^{-a-1})\right),
\end{aligned}
$$

which establishes (5.12). □

Note that the difference between (5.12) and Selçuk's formula (5.4) lies in the factor $\sqrt{1+\frac{N}{2^n}}$ of the term $\Phi^{-1}(1-2^{-a-1})$. Since Φ is monotonously increasing, our estimate for P_S is always smaller or equal to (5.4), and the resulting data complexity required for a certain advantage and P_S will always be at least as big as the one of (5.5).

The biggest deviations between both models occur when the influence of the second term $\sqrt{1+\frac{N}{2^n}}\cdot\Phi^{-1}(1-2^{-a-1})$ grows. This can happen if the adversary seeks a particularly big advantage a, or when the number of known plaintexts gets close to 2^n. Both cases typically occur when the cryptanalyst is aiming for the maximum possible number of rounds that can be broken by his respective linear attack.

5.2.3 Non-monotonicity of success rate as function of data complexity

Consider any fixed given combination of the bias ϵ, the block length n and the advantage a. The success probability of a linear attack is then a function of the number of known plaintexts N only and can hence be expressed as $P_S(N)$. Even though our estimate for $P_S(N)$ given by Theorem 5.10 is always smaller or equal to Selçuk's formula (5.4), the addition of the second term results in a function that is not necessarily monotonously increasing in N anymore.

Proposition 5.11. *For fixed ϵ, a and n, the success probability $P_S(N)$ with respect to the data complexity as given by Eq. (5.12) attains a relative maximum at*

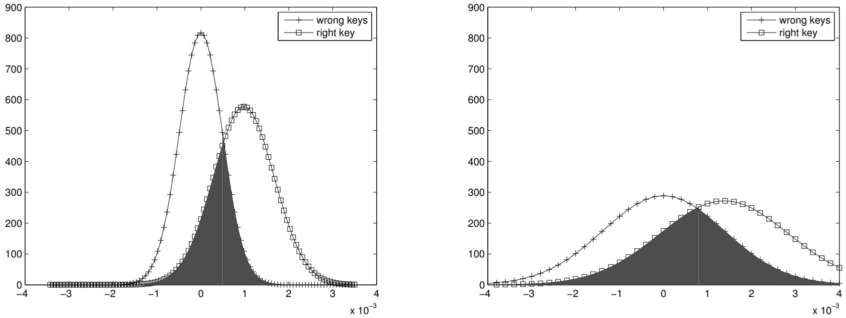
$$\widehat{N} \overset{\text{def}}{=} \frac{4|\epsilon|^2 \cdot 2^{2n}}{\left(\Phi^{-1}(1-2^{-a-1})\right)^2 - 4|\epsilon|^2 \cdot 2^{2n}}. \tag{5.14}$$

Proof. Elementary calculus shows that (5.12) has a stationary point at $\widehat{N}$ as defined above, and since its derivative changes sign from negative to positive at $\widehat{N}$, this stationary point is a relative maximum of $P_S(N)$. ☐

Proposition 5.11 implies that our model can in certain cases predict a decrease in success probability for an increased number of known plaintexts. While this may seem counterintuitive at first, one has to take into account that the success probability depends on the overlapping area between two approximately normal distributions, namely $\mathcal{N}(\epsilon, \frac{1}{4N})$ for the right key and $\mathcal{N}(0, \frac{1}{4}\left(\frac{1}{N} + \frac{1}{2^n}\right))$ for the wrong keys. In the context of small ϵ and large N of the order 2^n, increasing N can actually result in increasing the overlapping area, and hence decrease the success probability. An attack exploiting a fixed linear approximation with fewer known plaintexts could therefore be more efficient than with more given samples.

A given advantage a corresponds to a fixed threshold T for distinguishing the distributions, with the type I error $\mathcal{E}_I = 1 - P_S$ varying with N, and fixed type II error $\mathcal{E}_{II} = 2^{-a-1}$. Having $P_S(N) = P_S(N')$ for $N \neq N'$ is therefore equivalent to having the same overlapping area $\mathcal{E}_I + \mathcal{E}_{II}$ between the distributions for N and N' samples. This is depicted in Figure 5.5: the overlapping area $\mathcal{E}_I + \mathcal{E}_{II}$ between the two Gaussian distributions in Figure 5.5a and 5.5b is the same for different values of N.

We note that two conditions have to be fulfilled to be able to speak of meaningful (i.e., practically relevant) non-monotonous behaviour: First, the condition $\widehat{N} < 2^n$ has to be satisfied (since N cannot exceed 2^n); and second, one must have $P_S \geq 2^{-a}$, i.e. the success probability of the attack must be



(a) Situation for $N = 2^{20}$.

(b) Situation for $N = 2^{17}$.

Figure 5.5: Example of an equal overlapping area between $\mathcal{N}(\epsilon, \frac{1}{4N})$ and $\mathcal{N}(0, \frac{1}{4}\left(\frac{1}{N} + \frac{1}{2^n}\right))$ for $n = 20$ and different values of N.

higher than two times the false positive rate. Otherwise, the adversary would have to repeat the attack $1/P_S > 2^a$ times and gain only a bits advantage over the exhaustive search.

An example of a parameter combination fulfilling both conditions is $|\epsilon| = 2^{-10}, n = 20$ and $a = 12$, i.e., seeking a large advantage out of an approximation with only marginal bias. In this case, $\widehat{N} \approx 2^{18.75} < 2^{20}$, and $P_S(\widehat{N}) \approx 2^{-9.89} > 2^{-12} = 2^{-a}$. With $P_S(2^{20}) \approx 2^{-10.45}$, this constitutes a meaningful example where using more samples actually decreases the success probability. This theoretical prediction has been verified in the real world using experiments with SmallPresent-20. The recovery of 10000 keys exhibiting exactly the bias $\epsilon = 2^{-10}$ was attempted for different values of N. The results given in Figure 5.6 confirm the non-monotonous behaviour.

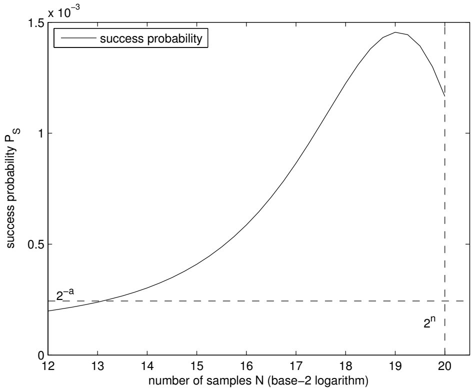
5.2.4 Evaluation of the data complexity

In practice, when evaluating a particular linear attack (where n and ϵ are fixed), it is often interesting to determine the number N of required known plaintexts for certain success probabilities and advantages that are sought by the attacker. In the case of $P_S = 1/2$ and an arbitrary fixed advantage of $a \geq 1$ bits, equation (5.12) yields a closed formula for N:

Corollary 5.12. *With the assumptions of Theorem 5.10, using a linear approximation with bias* $|\epsilon| > \Phi^{-1}(1 - 2^{-a-1})/2^{n/2-1}$*, the number* N *of known plaintexts required to obtain an advantage of* $a \geq 1$ *bits with success probability*

N	theoretical P_S	experimental P_S
2^{17}	0.00072	0.0006
2^{18}	0.00096	0.0009
$2^{18.5}$	0.00104	0.0009
$2^{18.75}$	0.00105	0.0011
2^{19}	0.00104	0.0010
$2^{19.5}$	0.00093	0.0009
2^{20}	0.00071	0.0007

(a) Experimental success probability.



(b) Plot of $P_S(N)$.

Figure 5.6: Experimental verification of non-monotonous behaviour for SmallPresent-20 with $\epsilon = 2^{-10}$ and $a = 12$.

$P_S = 1/2$ *is given by*

$$N \approx 1/\left(\left(2\epsilon/\Phi^{-1}(1-2^{-a-1})\right)^2 - 2^{-n}\right). \tag{5.15}$$

Proof. Note that for $P_S = 1/2$, $\Phi^{-1}(P_S) = 0$. The claim then follows directly from (5.12). □

The condition $|\epsilon| > \Phi^{-1}(1-2^{-a-1})/2^{n/2-1}$ in Corollary 5.12 basically prevents the estimate for N from becoming negative. This happens if the sought advantage a is too big for the given bias $|\epsilon|$, resulting in a data requirement of $N > 2^n$ texts, which is clearly impossible and a step outside the model.

For values of P_S different from $1/2$, we can determine N by means of an efficient numerical procedure for given $P_S, a, |\epsilon|$ and n. Note that this procedure is equally applicable to the case $P_S = 1/2$.

Proposition 5.13. *With the assumptions of Theorem 5.10, for fixed* ϵ, P_S, n *and* a*, the data complexity* N *can be determined numerically using Algorithm 5.1 up to an absolute error of* $1 - 2^{-n}$ *in linear time in the block length* n*.*

Proof. Denote $\widehat{N}$ as in (5.14) and write $N_{\text{max}} \overset{\text{def}}{=} \min\{\widehat{N}, 2^n\}$. It then follows from (5.14) that the function

$$N \mapsto 2\sqrt{N}\epsilon - \sqrt{1 + \frac{N}{2^n}}\Phi^{-1}(1 - 2^{-a-1})$$

is monotonously increasing on the interval $[1, N_{\text{max}}]$. We can hence apply bisection to find the value of N leading to this particular success probability $P_S(N)$ as in (5.12). In the case of non-monotonous behaviour of $P_S(N)$ on $[N_{\text{max}}, 2^n]$, this approach yields the optimal (lower) N resulting in the given success probability. The algorithm is detailed in Algorithm 5.1.

Denote by $l^{(i)}$ and $u^{(i)}$ the interval bounds after the i-th iteration. Then $u^{(i)} - l^{(i)} \leq \frac{2^n - 1}{2^{i-1}}$. Denoting the exact solution by $\tilde{N}$ and the i-th approximate solution as $N^{(i)} = \frac{l^{(i)} + u^{(i)}}{2}$, we have

$$|N^{(n)} - \tilde{N}| \leq \frac{u^{(n)} - l^{(n)}}{2} \leq \frac{2^n - 1}{2^n} = 1 - 2^{-n}$$

after n iterations. ☐

Algorithm 5.1 runs very efficiently even for large block sizes. For instance, a straightforward Matlab implementation computes the value $N = 2^{126.76}$ for $n = 128$, $|\epsilon| = 2^{-61.9}$, $a = 10$ and $P_S = 0.95$ in about 0.09 seconds on an Intel Core2 Duo E8400.

5.3 Experimental results

In this section, we summarise the results of experiments carried out to verify the accuracy of the estimate given by Theorem 5.10 and Proposition 5.13 and compare it to other models.

The experiments were first carried out on SmallPresent-20, a small-scale variant [108] of the block cipher present with block size $n = 20$ bits. The

Algorithm 5.1 Numerical computation of the data complexity N

```
Input: Bias $\epsilon$, block length $n$, success probability $P_S \geq 2^{-a}$, precision bound $\nu$ (*)
Output: Data complexity $N$ required for the given parameters.
 1: Define $f(N) = 2\sqrt{N}|\epsilon| - \sqrt{1 + \frac{N}{2^n}}\Phi^{-1}(1 - 2^{-a-1})$
 2: Calculate $\widehat{N} \leftarrow \left(4|\epsilon|^2 \cdot 2^{2n}\right) / \left(\left(\Phi^{-1}(1 - 2^{-a-1})\right)^2 - 4|\epsilon|^2 \cdot 2^{2n}\right)$
 3: $lower \leftarrow 1$, $upper \leftarrow \min\{\widehat{N}, 2^n\}$, $i \leftarrow 0$
 4: while $|f(lower) - P_S| > 10^{-\nu}$ and $i < n$ do
 5:     $mid \leftarrow \frac{lower+upper}{2}$
 6:     if $f(mid) < P_S$ then
 7:         $lower \leftarrow mid$
 8:     else
 9:         $upper \leftarrow mid$
10:     end if
11:     $i \leftarrow i + 1$
12: end while
13: return $lower$
```

*The value of ν in step 4 is used to early-abort fast-converging iterations as soon as an adequate precision is reached. A recommended value is $\nu = 15$.

original key schedule algorithm was used. In all experiments, we fixed a linear approximation with a certain bias ϵ and a success probability and then analysed the data complexity N which is required to obtain different levels of advantage with this P_S. Each experiment for a certain combination of N and a was averaged over 1000 times to obtain a reliable relation between N and a for this fixed P_S.

To verify the independence of our experimental findings from the structure of SmallPresent, all experiments were repeated with RC6-5/r/10, an instantiation of RC6 [145] with a block size of 20 bits and an 80-bit key. The results on this small-scale variant of RC6 indicate that our model is equally applicable to this substantially different block cipher structure.

5.3.1 Experiment 1: The impact of a low bias

In the first experiment on SmallPresent, a linear approximation with bias $\epsilon = 2^{-8.22}$ was used. The success probability was fixed at 95%. From (5.12), the influence of the new model for wrong keys is expected to manifest itself already for small advantages given this relatively high P_S and low ϵ (compared to the block length). The results are depicted in Figure 5.7. The curve with squares was obtained using Proposition 5.13 with an estimation of the hull bias averaged over 200 random keys. We can see that the experiments follow the theoretical prediction very closely. The difference to the estimate of [152] is also apparent as soon as $a \geq 6$. For $a = 11$, Selçuk's formula can result in an underestimation of N of factor two. The line with crosses represents the estimate based on Algorithm 1 of [30].

The results of an analogous experiment on RC6-5/8/10 are given in Figure 5.8.

5.3.2 Experiment 2: The impact of a high advantage

In fact, Figure 5.7 also reveals another important difference between the experimental reality and Selçuk's estimate: The estimate of [152] greatly overestimates the maximum advantage that can be obtained even when using the full codebook of $N = 2^n$ texts. In order to explore this in more detail, a second experiment on SmallPresent with a higher bias of $\epsilon = 2^{-8}$ and a bigger range of advantages was carried out. Again, the experiments match very well the prediction of Proposition 5.13 based on the hull bias averaged over 200 random keys. As expected from (5.12), the difference between [152] and our estimate is very small for advantages of up to 8 bits, but again increases to a factor of about two for $a = 15$. Most importantly, however, our estimate predicts the experimentally observed maximum advantage with 2^{20} texts and $P_S = 0.95$ much more precisely than previous work: While an advantage of 31 bits with the full codebook seemed possible according to [152], the experiments and the estimate of Proposition 5.13 limit the advantage to about 15 bits, which is a significant improvement. Again, the line with crosses plots the estimate based on Algorithm 1 of [30], which follows [152] very closely.

The results of an analogous experiment on RC6-5/6/10 can be found in Figure 5.10.

It should be noted that these experimental results do not contradict those in [152]: Selçuk's formula was experimentally verified with an approximation for 6 rounds of DES with bias $\epsilon = 1.95 \cdot 2^{-9}$, which is so high compared to the block length $n = 64$ that the influence of the distribution for the wrong keys is negligible.

Our experiments indicate that Theorem 5.10 and its derivatives are unlikely to decrease the precision in comparison to previous work, since our estimates are more realistic for large a and/or low ϵ but very close to Selçuk's for

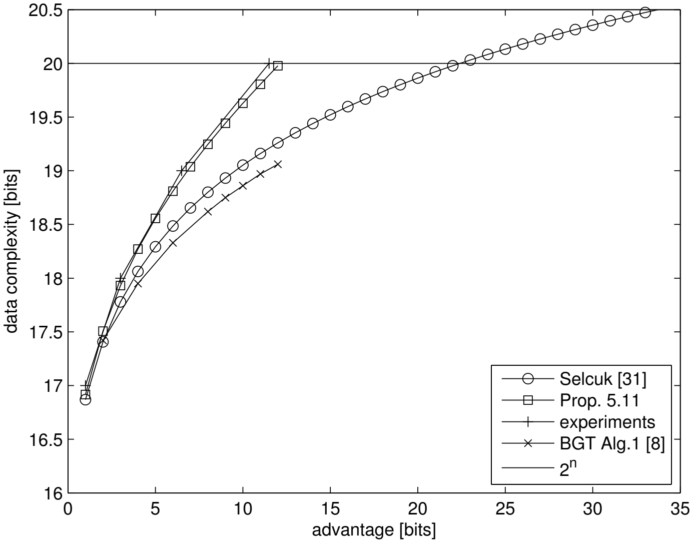


Figure 5.7: Theoretical and experimental evaluation of the data complexity estimate of Proposition 1 for different levels of advantage. The cipher is SMALLPRESENT with 8 rounds, $n = 20$, $|\epsilon| = 2^{-8.22}, P_S = 0.95$.

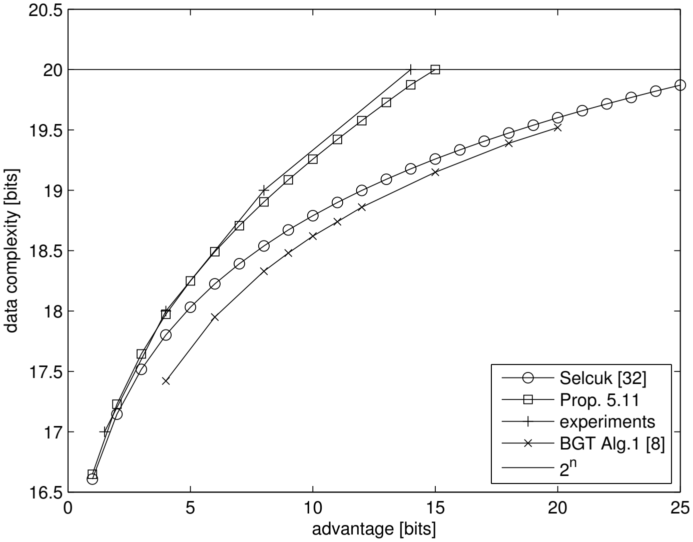


Figure 5.8: Theoretical and experimental evaluation of the data complexity estimate of Proposition 1 for RC6-5/8/10 and different levels of advantage. $n = 20$, $|\epsilon| = 2^{-8.09}, P_S = 0.95$.

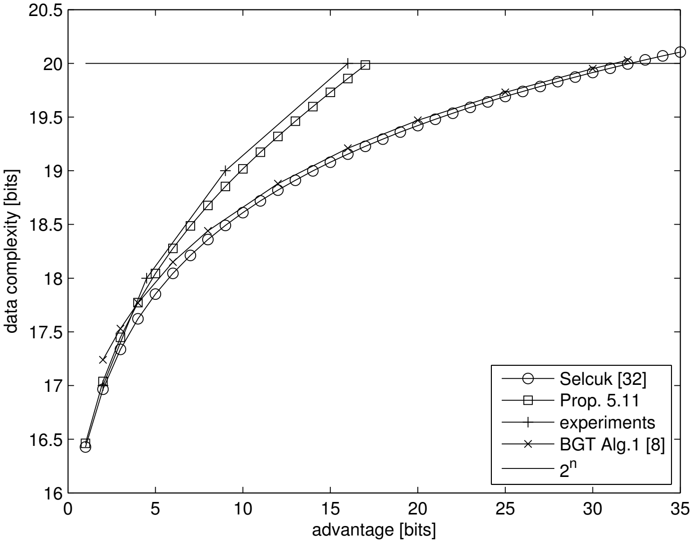


Figure 5.9: Theoretical and experimental evaluation of the data complexity estimate of Proposition 1 for higher levels of advantage. The cipher is SmallPresent with 8 rounds, $n = 20, \epsilon = 2^{-8}, P_S = 0.95$.

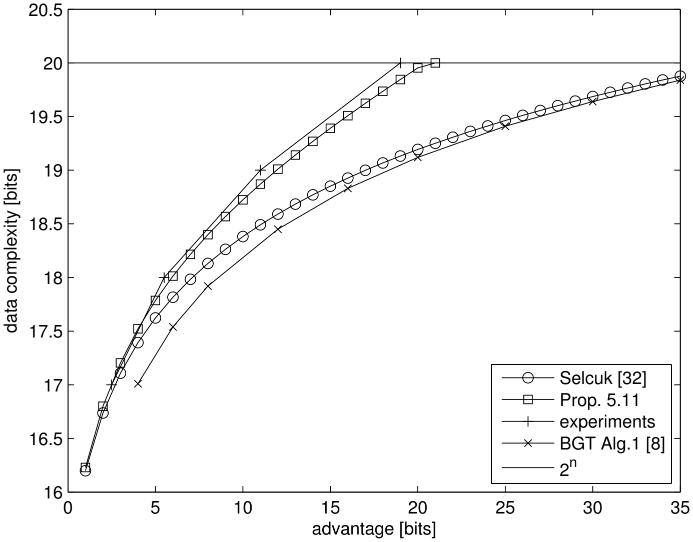


Figure 5.10: Theoretical and experimental evaluation of the data complexity estimate of Proposition 1 for RC6-5/6/10 and higher levels of advantage. $n = 20, \epsilon = 2^{-7.88}, P_S = 0.95$.

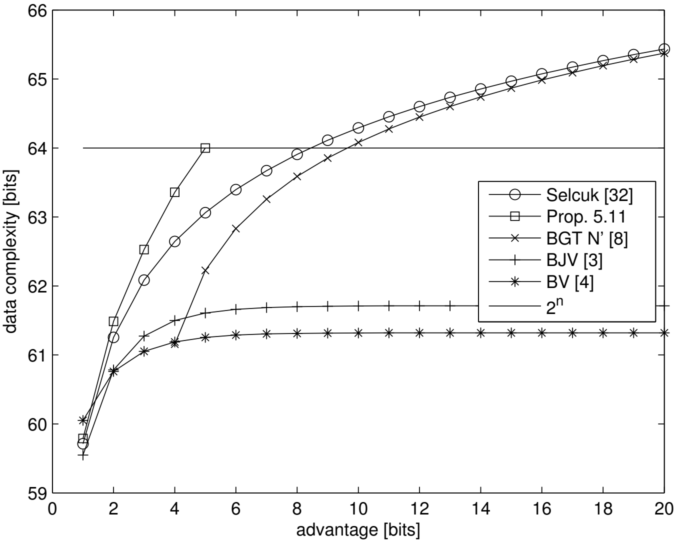


Figure 5.11: Evaluation of the data complexity estimate of Proposition 1 for bigger block length and marginal bias. $n = 64, \epsilon = 1.49 \cdot 2^{-32}, P_S = 0.5$.

small advantages and/or high biases. They can hence be used as a universal replacement.

5.3.3 Larger block sizes

Given the experimental evidence supporting the accuracy of the estimates based on Theorem 5.10, it remains to investigate the impact of the new model for larger block sizes where no practical experiments can be carried out.

The estimated data complexity corresponding to different levels of advantage for a $n = 64$ bit block length and a marginal bias of $\epsilon = 1.49 \cdot 2^{-32}$ is plotted in Figure 5.11. Again, even for a lower success probability of $P_S = 0.5$, differences of up to factor two occur between the estimate of Selçuk and this work. This figure also includes comparisons to the first-order estimate of [30] and the estimates (5.6) of [9] and (5.7) of [10].

Finally, we expect that increasing the bias improves the estimate of Selçuk for lower advantages, but may again result in an overestimate of the maximum achievable advantage. As shown in Figure 5.12, our estimate predicts a maximum advantage of 52 bits with the full codebook, while an advantage of 105 bits was allowed by Selçuk's estimate. Note also that the first-order estimate N' of [30] very rapidly converges to Selçuk's estimate here.

5.4 Impact of the linear hull effect

The general methodologies to evaluate the complexity of linear attacks at hand assume the exact bias or its good estimate is given to the adversary. Practically speaking, however, this is never the case for almost any real-world cipher. This is due to the fact that for a relatively large block size (e.g. longer than 50 bits) it is challenging to exactly evaluate the bias even for one fixed known key. That is why most linear attacks base their complexity estimates on one or several known characteristics (rather than on the entire linear hull bias). However, a systematic methodology to estimate the complexity of Matsui's Algorithm 2 based on a group of dominant characteristics rather than on the full linear hull appears to be still lacking.

To address this, we propose a technique that takes the linear hull effect into account. In this context, we make two observations. First, we propose to split the linear hull into a signal part and a noise part. The signal part is then sampled for random cipher keys to obtain a more reliable evaluation of the impact of those trails. Second, we statistically model the noise part to make the estimation of complexity more realistic.

5.4.1 Motivation

In order to accurately estimate the complexity of a linear attack, regardless whether with the methods listed in Section 5.1.3 or with Corollary 5.12 and Proposition 5.13, one has to know the (absolute value of) the bias ϵ_r of the used linear approximation (α, β) for the right key k_r.

Generally speaking, for a block cipher, it may not be feasible to evaluate ϵ_r for any given key, even if it is known to the adversary (which is not the case for the *unknown* k_r to be determined in a linear attack). However, for some classes of ciphers with structure, the situation is slightly different. For key-alternating ciphers and some Feistel ciphers which can be written as key-alternating ciphers (see Section 2.2.2), the *absolute value of the bias* of each linear trail is independent of the key; only the sign depends on the key value. Hence, the value of bias for each trail can be efficiently evaluated for any key, once the trail is known. Now, if the approximation (α, β) contains only one

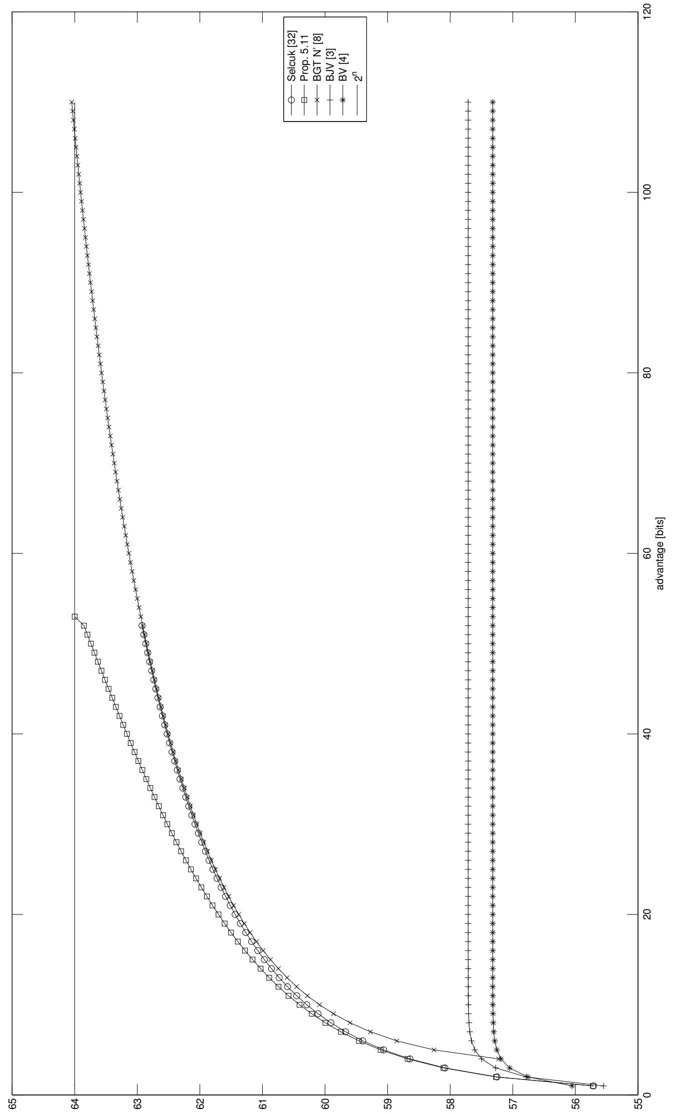


Figure 5.12: Evaluation of the data complexity estimate of Proposition 1 for bigger block length and higher levels of advantage. $n = 64, \epsilon = 1.49 \cdot 2^{-30}, P_S = 0.5$

trail, the exact value of $|\epsilon_r|$ is known and identical for any k_r. Therefore, the success probability and data complexity estimates are equally valid for all keys.

In most ciphers, however, practically interesting hulls (α, β) will contain more than one trail, especially over many rounds. For realistic block lengths and number of rounds, it is usually infeasible to compute the complete hull (i.e., enumerate all its trails). In contrast to a hull with just one trail, the contribution of the trail bias to the absolute hull bias now depends on the key even for key-alternating ciphers. This is known as the *linear hull effect*, which causes the trail biases to be added or subtracted depending on the value of the key [86, 109, 125, 147]. As a consequence, even for key-alternating ciphers, it usually becomes infeasible to compute the exact value of $|\epsilon_r|$ or even its average over all keys.

Once the right key k_r is fixed, the adversary faces an unknown fixed value of ϵ_r. Since ϵ_r varies from one value of k_r to another, this yields a distribution of ϵ_r over the values of the right key k_r. In this connection, the question arises *which value of ϵ_r to take for the evaluation of the attack complexity.*

For instance, the work [139] fixes the expanded key of PRESENT to zero to estimate ϵ_r. However, we note that taking a zero key as the reference point is exactly as justified as choosing a random key for this purpose. And different keys will provide different estimations of ϵ_r. Moreover, the key schedule of PRESENT arguably does not allow for the zero expanded key. Though we do believe that the resulting complexity claims of [139] for PRESENT are adequate, one can actually estimate ϵ_r more accurately. Furthermore, for a cipher other than PRESENT, fixing the expanded key to zero might be misleading in terms of complexity estimates.

5.4.2 Decomposing linear hulls: signal and noise

To address these problems for key-alternating ciphers, we propose a two-fold approach. First, we split ϵ_r into a signal part (resulting from the known trails with high contribution) and a noise part (to account for the unknown remainder of the hull). Note that previous approaches omitted the influence of the unknown trails completely and as has been demonstrated [109, 125], the influence of the unknown part of the hull can be very significant. Second, we average the data complexity estimate over a number of randomly drawn master keys, as opposed to fixing one specific expanded key.

We now detail this procedure that aims at taking the impact of the linear hull effect into account. Let (α, β) be the linear approximation under consideration. Assume that only one or a small number of trails with high contribution to the absolute bias of the hull (α, β) are known. In a linear attack, to recover (part of) the key k_r, this hull has an unknown bias $\epsilon_r(k_r)$, potentially varying from key to key. For each fixed value of the key, the actual value of $\epsilon_r(k_r)$ can be decomposed into the contributions that stem from the biases of the known trails and the biases of the remaining unknown trails in the hull. We define the former to be the *signal* and the latter the *noise*:

Definition 5.14. Consider the linear approximation (α, β) over an R-round iterative block cipher and a fixed cipher key k_r. The bias $\epsilon_r(k_r)$ of the hull (α, β) is given by

$$\epsilon_r(k_r) = \sum_{u_0=\alpha, u_R=\beta} \epsilon_U(k_r),$$

with $\epsilon_U(k_r)$ denoting the bias of the trail U with key k_r. Suppose some t dominant trails $\mathcal{U} = \{U_1, \ldots, U_t\}$ of the hull (α, β) are known. By defining

$$\epsilon_{U_{\text{signal}}}(k_r) \stackrel{\text{def}}{=} \sum_{U \in \mathcal{U}} \epsilon_U(k_r) \tag{5.16}$$

$$\epsilon_{U_{\text{noise}}}(k_r) \stackrel{\text{def}}{=} \sum_{(\alpha,\beta)\setminus\mathcal{U}} \epsilon_U(k_r), \tag{5.17}$$

we obtain a repartitioning of the above sum as follows:

$$\epsilon_r(k_r) = \epsilon_{U_{\text{signal}}}(k_r) + \epsilon_{U_{\text{noise}}}(k_r). \tag{5.18}$$

Based on Corollary 5.8, the noise part $\epsilon_{U_{\text{noise}}}(k_r)$ of the trail contributions can now be modeled to approximately follow a normal distribution $\mathcal{N}(0, 2^{-n-2})$ over the right keys. This leads to a natural approach to estimate $|\epsilon_r(k_r)|$: Sample and average $N(k_r)$ computed with (5.18) over a number of keys k_r, taking the noise from the unknown part of the hull into account by drawing it from $\mathcal{N}(0, 2^{-n-2})$ afresh for each tried key. For computing the signal part, the relation between the expanded key and the trail contributions has to be efficiently computable.

The latter is the case for the numerous class of key-alternating ciphers. We give an explicit procedure for key-alternating ciphers in the next subsection.

5.4.3 An explicit algorithm for key-alternating ciphers

We consider a linear approximation (α, β) over r rounds. Let K denote the key expansion of the key k. In the case of key-alternating ciphers (or Feistel ciphers which can be written as such), the contributions of the biases of the individual trails to the bias of the hull for a fixed key k can be explicitly computed [54,57] as:

$$\epsilon(k) = \sum_{u_0=\alpha, u_r=\beta} (-1)^{d_U \oplus U^T K} |\epsilon_U|. \tag{5.19}$$

This leads to the following algorithm for estimating the data complexity N of a linear attack on an n-bit block cipher using the linear approximation (α, β): We know t trails from the hull and sample $\epsilon_{U_{\text{signal}}}$ over a number of keys by means of (5.19), each time adding $\epsilon_{U_{\text{noise}}}$ sampled from $\mathcal{N}(0, 2^{-n-2})$. For each tried key, we compute an estimate for N based on this value of ϵ_r. Then the average over all tried keys is taken as the final estimate for N. This procedure is described in Algorithm 5.2.

Algorithm 5.2 Computation of N using the signal-noise decomposition of the hull for key-alternating ciphers.

```
Input: Trails $U_j, 1 \le j \le t$ from the hull $(\alpha, \beta)$, their absolute biases $|\epsilon_{U_j}|$,
    number of keys $\ell$ to sample.
Input: Block length $n$, success probability $P_S \ge 2^{-a}$.
Output: Estimate of the data complexity $N$ required for the given parameters.
for $i = 1, \ldots, \ell$ do
    Select the master key $k_i$ uniformly at random and compute the expanded key.
    Sample $noise(k_i)$ from $\mathcal{N}(0, 2^{-n-2})$.
    Compute
        $\epsilon(k_i) = \epsilon_{U_{\text{signal}}}(k_i) + \epsilon_{U_{\text{noise}}}(k_i)$
        $= \sum_{j=1}^{t} (-1)^{d_{U_j} \oplus U_j^T K_i} |\epsilon_{U_j}| + noise(k_i).$
    Compute $N(k_i)$ based on $\epsilon(k_i)$ with Algorithm 5.1.
end for
return Average $\overline{N} = \frac{1}{\ell} \sum_{i=1}^{\ell} N(k_i)$.
```

5.4.4 Experimenting the signal/noise decomposition

We have performed experiments on SMALLPRESENT-20 to illustrate the effect of the signal/noise decomposition of a linear hull. With a block length of $n = 20$ bits, and an 80-bit key space, it is not feasible to compute the exact distribution or even only the exact average bias of a hull (α, β) over the keys. Since n is small, sampling and averaging over some keys is possible here, but this is not the case anymore for realistic block lengths.

Consider the hull $(\alpha, \beta) = (0x20400, 0x20000)$ over 3 rounds. A branch-and-bound search for trails with $|\epsilon| \geq 2^{-11}$ yields 8 trails from the hull: three with absolute bias $|\epsilon| = 2^{-10}$ and five with $|\epsilon| = 2^{-11}$. Based on this data, the following estimates for the data complexities of a linear attack with $P_S = 0.95$ and varying advantages were computed based on Proposition 5.13:

1. N for $\epsilon_r(k_r)$ of the known trails for one cipher key k_r;
2. N determined with Algorithm 5.2 with $\kappa = 200$ keys, but without the noise part;
3. N determined with Algorithm 5.2 with $\kappa = 200$ keys;
4. N for an estimation of the hull bias averaged over 200 random keys.

Additionally, the actual data complexity was determined experimentally. Each experiment for a certain combination of N and a was averaged over 1000 times to obtain a reliable relation between N and a for this fixed P_S.

The results are depicted in Figure 5.13. One observes that summing the trail biases for one key results in a far too optimistic estimation. Averaging the data complexity estimates for the signal trails for 200 keys (but without the noise part) improves the accuracy, but yields an overestimate here. This can be attributed to the impact of two factors: First, the hull must contain more signal trails that are missing in our set of eight trails; and second, the noise impact of the remainder of the hull is not accounted for. Additionally taking the noise into account yields a more realistic estimate. In this specific case though, it is still an overestimate since here obviously the remainder of the hull constructively helps to increase the bias for many keys.

We also observe that the average of the complete hull bias over 200 keys quite accurately matches the biases that occurred in the experiments, but note that this estimation is available only due to the small block length and will not be available for most realistic block sizes. In the latter case, the experiments suggest that our complexity estimate with sampling the signal part and modelling the noise part of the hull still appears more adequate than estimates not considering the linear hull effect.

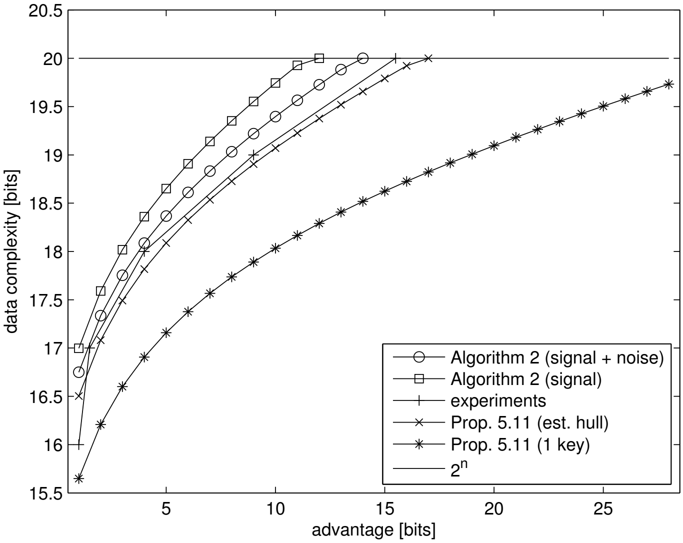


Figure 5.13: Theoretical and experimental evaluation of the data complexity with the signal-noise decomposition of Alg 2. Cipher is SmallPresent with 5 rounds, $n = 20, P_S = 0.95$, the signal part contains 8 trails U_j with $2^{-10} \leq |\epsilon_{U_j}| \leq 2^{-11}$. Experimental value of ϵ is $2^{-8.02}$.

5.5 Analysing the complexity of differential attacks with structures

As noted in the introduction to this chapter, comprehensive models for the analysis of extended variants of basic linear and differential attacks are still scarce. The work [30] provides a first step in this direction, albeit focusing more on obtaining a unified framework for many extensions at the expense of accuracy for each single variant. Furthermore, the estimates obtained in [30] are asymptotic and as such cannot be used universally in practice.

5.5.1 Motivation: Using multiple differentials in an attack

Multiple differential cryptanalysis is an extension of basic differential cryptanalysis. Instead of a single differential (δ_a, δ_b), it makes use of multiple differentials with potentially more than one input and more than one output difference. Since a given chosen plaintext pair is more likely to follow one of multiple differentials, there is a higher probability of hitting a right pair for one of them

$$\begin{tikzcd}
x \arrow[r, leftrightarrow] \arrow[d, leftrightarrow] \arrow[dr, leftrightarrow, dashed] & x \oplus \Delta_0^1 \arrow[d, leftrightarrow] \arrow[dl, leftrightarrow, dashed] \\
x \oplus \Delta_0^2 \arrow[r, leftrightarrow] & x \oplus \Delta_0^1 \oplus \Delta_0^2
\end{tikzcd}$$

Figure 5.14: Pairs of plaintext organised in a quartet.

than in a standard differential attack. This can reduce the data complexity provided that the computational complexity of processing the set of all possible pairs arising from multiple input and output differences can be controlled in an efficient manner.

Historically, the idea of using more than one differential in an attack was introduced in the restricted form of *quartets* in the original paper on differential cryptanalysis [22]. A quartet is a structure of four chosen plaintexts simultaneously containing two pairs with an input difference Δ_0^1 and two pairs with an input difference $\Delta_0^2 \neq \Delta_0^1$. In this way, four pairs satisfying two input differences can be derived from only four chosen plaintexts.

This has later been extended to "octets", and so on. The natural generalisation of this concept is called a *structure* [20,29], which provides an economic way of organising the pairs for multiple input differences:

Definition 5.15. Let $\{\Delta_0^1, \ldots, \Delta_0^t\}$ be a set of t input differences. For an arbitrary but fixed value of x, a collection of plaintexts of the form

$$\bigcup \{x \oplus \Delta \mid \Delta \in \mathrm{span}\{\Delta_0^1, \ldots, \Delta_0^t\}\}, \tag{5.20}$$

with span denoting the linear span operator, is called a *structure.*

We note that a more generalised type of difference (e.g., truncated differentials [102]) can be used in the setting of Definition 5.15. An example of a structure with $t = 2$ (hence, a quartet) is given in Figure 5.14. Note that besides the two pairs with the original input differences Δ_0^1 and Δ_0^2, it implicitly contains another two pairs with input difference $\Delta_0^1 \oplus \Delta_0^2$ from $\mathrm{span}\{\Delta_0^1, \Delta_0^2\}$.

The concept of a structure can in principle be used in a general multiple differential attack with multiple input and output differences. It is however most efficient when used with a single output difference [20]. We therefore restrict our discussion of differential attacks using structures to the case where multiple input differences, but a single output difference are used. We call such attacks *structure attacks* [161].

Combining multiple differentials in a structure attack can obviously reduce the data complexity compared to a classic single differential attack. In order to reduce the overall time complexity, however, the differentials have to be chosen in a careful way. Furthermore, it is usually not obvious a priori which choice of differentials can actually lead to an improvement. In Section 5.6, we develop a model that provides some guidance in this direction.

5.5.2 Previous work

A notable exception to the apparent lack of study of the complexity of extensions of standard linear and differential cryptanalysis is the framework for the analysis of multiple differential cryptanalysis by Blondeau and Gérard [27, 29]. This paper provides an explicit formula to compute the success probability of multiple differential cryptanalysis. Traditionally, a normal approximation to the binomial distribution was used to evaluate the success probability of a differential attack [152, 153]. However, this approximation is not very tight for typical parameters in differential cryptanalysis, where the success probability of a single Bernoulli experiment is very close to zero [154]. Then approximating the sum of multiple differentials by a normal distribution, this issue becomes even worse.

Blondeau and Gérard demonstrate that Selçuk's method cannot be applied to multiple differential cryptanalysis and express the distribution of key counters instead in terms of a hybrid distribution including the Kullback-Leibler divergence and a Poisson distribution [29]. Blondeau and Gérard obtain the following formula for the success probability P_S:

$$P_S \approx 1 - G_*[G^{-1}(1 - \frac{l-1}{2^{n_k}-2}) - 1], \tag{5.21}$$

where n_k is the number of key candidates, l is the size of the list of suggested key candidates, and G^{-1} is defined by $G^{-1}(y) = \min\{x | G(x) \geq y\}$. The functions G and G_* are defined as follows:

$$G_*(\tau) \stackrel{\text{def}}{=} G(\tau, p_*) \quad \text{and} \quad G(\tau) \stackrel{\text{def}}{=} G(\tau, p),$$

where $p_* = \frac{\sum_{i,j} p_*^{(i,j)}}{|\Delta_0|}$ and $p = \frac{|\Delta|}{2^n |\Delta_0|}$. $p_*^{(i,j)}$ is the probability for the differential with the i-th input difference value and the j-th output difference value, n is the block size, $|\Delta_0|$ is the number of input difference values and $|\Delta|$ is the number of differentials. $G(\tau, p_*)$ and $G(\tau, p)$ can be calculated with the following equations:

$$G(\tau, q) \stackrel{\text{def}}{=} \begin{cases} G_-(\tau, q) & \text{if} \quad \tau < q - 3 \cdot \sqrt{q/N_s}, \\ 1 - G_+(\tau, q) & \text{if} \quad \tau > q + 3 \cdot \sqrt{q/N_s}, \\ G_{\mathcal{P}}(\tau, q) & \text{otherwise}, \end{cases} \tag{5.22}$$

where $G_{\mathcal{P}}(\tau, q)$ is the cumulative distribution function of the Poisson distribution with parameter qN_s, and N_s is the number of samples. $G_-(\tau, q)$ and $G_+(\tau, q)$ are defined as follows:

$$G_-(\tau, q) \stackrel{\text{def}}{=} e^{-N_s D(\tau \| q)} \cdot [\frac{q\sqrt{1-\tau}}{(q-\tau)\sqrt{2\pi\tau N_s}} + \frac{1}{\sqrt{8\pi\tau N_s}}], \tag{5.23}$$

$$G_+(\tau, q) \stackrel{\text{def}}{=} e^{-N_s D(\tau \| q)} \cdot [\frac{(1-q)\sqrt{\tau}}{(\tau-q)\sqrt{2\pi N_s(1-\tau)}} + \frac{1}{\sqrt{8\pi\tau N_s}}], \tag{5.24}$$

where $D(\tau \| q)$ is the Kullback-Leibler divergence between two binomial distributions with respective success probilities p and q which is defined by:

$$D(\tau \| q) \stackrel{\text{def}}{=} \tau \ln\left(\frac{\tau}{q}\right) + (1-\tau)\ln\left(\frac{1-\tau}{1-q}\right). \tag{5.25}$$

On the assumptions for this analysis

In the analysis of the key counters in multiple differential cryptanalysis [29], a seemingly trivial complication arises: For a fixed key, when summing over all possible values of x, both x and $x \oplus \Delta_0^i$ will be counted for any possible input difference Δ_0^i. In a standard differential attack with a single differential, the set of plaintexts is implicitly partitioned in two disjoint sets by the structure of the pairs. In the case of multiple differentials, this becomes nontrivial [29].

Therefore, in order to guarantee that each pair is counted only once in their analysis, Blondeau and Gérard give Definition 5.16 as a necessary condition on the set of the input differences Δ_0:

Definition 5.16. The set of input differences Δ_0 is admissible if there exists a set χ of $N/2$ plaintexts that fulfils the condition:

$$\forall \delta_0^{(i)} \in \Delta_0, \forall x \in \chi, x \oplus \delta_0^{(i)} \notin \chi, \tag{5.26}$$

where N is the number of chosen plaintexts.

However, this condition imposes a severe restriction on the freedom with which the attacker can select suitable differentials. As a result, many potentially valuable differentials have to be excluded. For example, independent of the algorithm under consideration, the set of input differences $\Delta_0 = \{1_x, 2_x, 3_x\}$ is never admissible in any substitution-permutation network because of this condition, since the overlapping bits of $3_x = 1_x \oplus 2_x$ will always result in double-counting.

5.6 A Model for structure attacks

Compared to approaches based on a normal approximation of Binomial distributions, the method of [29] outlined in the previous section provides a more accurate estimation of the success probability. Since structure attacks are a special case of multiple differential cryptanalysis, these results also immediately apply to them.

Due to the limitations implied by condition 5.16, it remains to investigate whether they can be mitigated in a specialised model that is tailored for the subclass of structure attacks.

5.6.1 Principle of the attack

The structure attack is a form of differential cryptanalysis which uses multiple input differences and a single output difference. Structure attacks are a special case of multiple differential cryptanalysis, but having only one output difference and organising the chosen plaintexts in structures allows for a dedicated attack procedure, which we describe in this section.

A structure attack is performed in three phases [161]:

1. **Data Collection Phase:** Collect a large number of ciphertext pairs with the differences produced from the output difference of the differentials and the corresponding plaintext differences belong to the set of the input differences.
2. **Distillation and Data Analysis Phase:** Extract the differential statistic from the data and derive the list of the best candidates for some key bits from the collected ciphertext pairs.
3. **Key Search Phase:** Search the list of candidates and all the remaining master key bits (*i.e.*, the unexpanded key from which the round subkeys are derived).

For the first two phases, we can leverage structures to obtain the expected number of right pairs with lower data complexity compared to a single differential. Now we will give a model to choose the differentials to reduce the complexity. For clarity of exposition, we describe the model for the case of a substitution-permutation network (SPN); however, the concept can analogously be applied to other block cipher constructions, most importantly Feistel ciphers.

Consider the following setting: We attack an R-round block cipher with a block length of n bits and a k-bit (master) key using $|\Delta_0|$ r-round differentials with a single output difference and multiple input differences, we denote these differentials as follows:

$$\Delta_0^i \xrightarrow{r} \Delta_r, \quad \text{with } p_i \overset{\text{def}}{=} \Pr(\Delta_0^i \xrightarrow{r} \Delta_r),\ (1 \leq i \leq |\Delta_0|), \tag{5.27}$$

where Δ_0^i and Δ_r are the i-th input difference and the output difference, respectively. When dealing with SPNs, it turns out to be advantageous to first use truncated differentials for the construction of the structures and then use a set of concrete differences during the attack.

We denote the number of plaintexts bits involved in the *active* S-boxes in the first round in any one of the differentials as N_p and the number of ciphertexts bits involved in the *non-active* S-boxes in the last round R by N_c. The number of distinct structures used in the attack is $2^{N_{st}}$, and the size of the key candidate list produced in phase 2 of the attack is denoted by ℓ. The amount of subkey bits guessed for partially decrypting the last $R-r$ rounds is denoted n_k.

In the attack, $2^{N_{st}}$ structures are constructed. In each structure, all the input bits to non-active S-boxes in the first round are fixed to some random value, while N_p input bits of all active S-boxes take on all 2^{N_p} possible values. Consequently, we ask for the encryptions of $2^{N_{st}+N_p}$ chosen plaintexts and there are $2^{N_{st}} \cdot 2^{N_p-1} = 2^{N_{st}+N_p-1}$ pairs for each differential. It is expected that about $2^{N_{st}+N_p-1} \cdot \sum_{i=1}^{|\Delta_0|} p_i$ of these pairs will produce the sought output difference Δ_r. These pairs are the right pairs needed to distinguish the correct from the wrong key guesses.

The attack can be described as follows. It is structurally very similar and can be seen as a generalisation of the differential attack on Serpent in [20].

1. For each structure:
    (a) Insert all the ciphertexts into a hash table indexed by the N_c bits corresponding to the non-active S-boxes in the last round.
    (b) For each entry with the same N_c bits value, check whether the input difference is any one of the total $|\Delta_0|$ possible input differences. If a pair satisfies one input difference, then go to the next step.

(c) For the pairs in each entry, check whether the output differences of the active S-boxes in the last round is compatible with the input differences resulting from Δ_r. If the pair passes the test, then go to the next step.

(d) Guess n_k bits subkeys to decrypt the ciphertext pairs to round r and check whether the obtained output difference at round r is equal to Δ_r. If so, add one to the corresponding counter.

2. Choose the list of the ℓ best subkey candidates from the counters.
3. For each of the ℓ subkey candidates: Guess the remaining master key bits, verify against one or two plaintext/ciphertext pairs.

5.6.2 Analysis of the attack

We now analyse the time and memory complexities of this attack.

Obviously, the time complexity of step 2 is negligible, so we denote by T_a, T_b, T_c, T_d and T_3 the time complexities of steps (a), (b), (c), (d) and 3, respectively. We make the further simplifying assumption that there are n_k independent subkey bits from the key schedule. If this is not fulfilled, the complexity of the key search phase can increase.

In step (a), we need to store all $2^{N_{st}} \cdot 2^{N_p}$ ciphertexts, accounting for $T_a = 2^{N_{st}+N_p}$ memory accesses. In step (b), we can assume that the ciphertexts behave as random n-bit values, implying we have to check about $2^{N_{st}+2N_p-N_c}$ hash table entries for collisions, out of which there are $2^{N_{st}+N_p-N_c}$ remaining to be analysed for compatibility with Δ_0 in step (c). Therefore, $T_b = 2^{N_{st}+2N_p-N_c}$ and $T_c = |\Delta_0| \cdot 2^{N_{st}+N_p-N_c}$ memory accesses. For step (d), the 2^{n_k} repetitions for the subkey guessing are to be balanced against the filter probability according to the output differences. T_d can therefore be approximated by $|\Delta_0| \cdot 2^{N_{st}+N_p-N_c}$ partial decryptions, leaving a complexity of $T_3 = \ell \cdot 2^{k-n_k}$ for the key search phase.

Since $|\Delta_0| \ll 2^{N_p}$, we have $T_c < T_b$, and the overall time complexity can consequently be expressed as follows:

$$T_a + T_b + T_c + T_d + T_3 \approx \begin{cases} T_a + T_3 & \text{if } N_p < N_c, \\ T_b + T_3 & \text{if } N_p > N_c, \\ 2T_a + T_3 = 2T_b + T_3 & \text{if } N_p = N_c. \end{cases} \tag{5.28}$$

This expression now provides some guidance with regard to the question how to choose the set of differentials, and in particular how many of them to afford.

If the time complexity in the key searching process T_3 is much smaller than the time complexity of the data collection process and the data analysis process, we can take $N_p = N_c$ to minimise the whole time complexity as the minimum value $2T_a$. Otherwise, we can try to take a larger value for N_p to increase the sum of the probabilities for differentials to further reduce the data complexity.

The latter case for instance applies in an attack scenario where the probability of many differentials are close to 2^{-n}, implying a low success rate P_S. Therefore, a large value for ℓ has to be chosen, which causes the complexity T_3 of step 3 to increase. In this case, increasing the number of input differences (and hence N_p) can help improving the attack, whereas increasing the number of output differences would not have this effect in the case of multiple differential cryptanalysis.

If, on the other hand, the probabilities of the differentials are much larger than 2^{-n}, we can choose more differentials without affecting the success probability. In order to minimize the time complexity, we will aim for $N_p = N_c$ according to our model.

5.6.3 Evaluating the success probability of structure attacks

Being a subclass of multiple differential attacks, the success probability P_S of a structure attack can be evaluated within the framework of Blondeau and Gérard described in Section 5.5.2, provided that the set of input differences fulfills condition 5.16. This is required to avoid counting both x and $x \oplus \delta_0^{(i)}$ for any $\delta_0^{(i)} \in \Delta_0$, i.e. guarantee that $N_s = N|\Delta_0|/2$ with N_s denoting the number of samples derived from N chosen plaintexts as in Section 5.5.2. Note that in this context, imposing condition 5.16 is *sufficient* for satisfying the intrinsic requirement on N_s, but not *necessary.* It enforces avoiding double counting *per differential*, while any other method of avoiding duplicates *globally* on the set of all pairs would suffice just as well.

It turns out, that since in structure attacks we only deal with one possible output difference, we can satisfy this condition on N_s even for sets of input differences violating condition 5.16: In our setting, $N = 2^{N_{st}+N_p}$, and the hash table will produce $N/2$ plaintext pairs with one input difference from N plaintexts, in total therefore $N_s = |\Delta_0|N/2$ plaintext pairs with $|\Delta_0|$ input diffference values. For structure attacks, the complexity analysis of [29] is therefore applicable independent of Def. 5.16.

This has additionally been verified by experiments on SmallPresent with block length of 24 bits, 12 rounds, and a set of 11 differentials with input differences violating Definition 5.16 and a single output difference.

We stress that this condition and the general model of [29] are still necessary for the analysis of the general case where one has multiple input and multiple output differences.

5.6.4 Ratio of weak keys in a structure attack

In general, the differential probability is related to the value of the key. As we use multiple differentials in the structure attack, we need to consider the ratio of keys which can produce the expected number of right pairs. We call those keys *weak keys* since the attacks are only expected to work for those.

Recall that a cipher is called *key-alternating* if it consists of an alternating sequence of unkeyed rounds and simple bitwise key additions (see Section 2.2.2). The fixed-key cardinality of a differential, denoted $N[K](a, b)$, is the number of pairs with input difference a and output difference b where the key K is fixed to a specific value. In [58, 60], Daemen and Rijmen give the following theorem.

Theorem 5.17. *Assuming that the set of pairs following a characteristic for a given key can be modeled by a sampling process, the fixed-key cardinality of a differential in a key-alternating cipher is a stochastic variable with the following distribution:*

$$\Pr(N[K](a, b) = i) \approx \text{Poisson}(i, 2^{n-1} EDP(a, b)),$$

where n is the block length, $EDP(a, b)$ denotes the expected differential probability of the differential (a, b), and the distribution function measures the probability over all possible values of the key and all possible choices of the key schedule.

For multiple differentials with multiple input differences and a single output difference, we have $p_j = \text{EDP}(a_j, b), 1 \leq j \leq |\Delta_0|$. We denote the fixed-key cardinality of multiple differentials (a_j, b) with a single output difference b by $N[K]\big\{(a_j, b)\big\}_j$. Based on Theorem 1, we can now derive

Theorem 5.18. *Under the assumptions of Theorem 5.17, in a key-alternating cipher, the fixed-key cardinality of multiple differentials with a single output difference is a stochastic variable with the following distribution:*

$$\Pr\Big(N[K]\big\{(a_j, b)\big\}_j = i\Big) \approx \text{Poisson}(i, 2^{n-1} \sum_j \text{EDP}(a_j, b)),$$

where the distribution function measures the probability over all possible values of the key and all possible choices of the key schedule.

Proof. The cardinality of multiple differentials equals the sum of the cardinalities of each differential (a_j, b) for the iterative cipher, so we have

$$N[K]\big\{(a_j, b)\big\}_j = \sum_j N[K](a_j, b).$$

From Theorem 1, the cardinality for each differential (a_j, b) has Poisson distribution. Making the standard assumption that the cardinalities of the differentials are independent random variables, the sum still is Poisson distributed with as λ-parameter the sum of the λ-parameters of the terms:

$$\lambda = \sum_j 2^{m-1}\mathrm{EDP}(a_j, b). \qquad \square$$

From Theorem 2, in a structure attack based on the differentials

$$\Delta_0^i \xrightarrow{r} \Delta_r, \quad \text{with } p_i \overset{\text{def}}{=} \Pr(\Delta_0^i \xrightarrow{r} \Delta_r),\ (1 \leq i \leq |\Delta_0|),$$

the ratio of the weak keys r_w that can produce more than or equal to μ right pairs can be computed as summing a tail of the above-mentioned Poisson distribution:

$$r_w \overset{\text{def}}{=} 1 - \sum_{x=0}^{\mu-1} \mathrm{Poisson}(x, 2^{m-1} \sum_{j=1}^{|\Delta_0|} p_i).$$

Note that when evaluating the ratio of weak keys, we have a different setting than when dealing with the distribution of the counters in a (multiple) differential attack. While approximating the distribution of the counters with either normal or Poisson distributions was shown to be problematic for accurately estimating the tails [29, 152], the distribution of the weak keys instead depends on the *cardinality* of the multiple differentials. In this setting, using the Poisson distribution as in Theorem 5.18 also yields a good approximation for the tails. This was also experimentally verified with small-scale variants of the block cipher PRESENT [108], with block lengths ranging from 8 to 24 bits.

Additionally, the accuracy of the weak key ratio r_w based on Theorem 5.18 has been verified by experiments on SmallPresent with a block length of 24 bits, 12 rounds and an master key with 8 bit entropy. 7 differentials with 7 different input and a single output difference were used. The λ-parameter of the Poisson distribution was $2^{23} \cdot \big(5 \cdot 2^{-23} + 2 \cdot 2^{-22}\big) = 2^{3.17}$. The distribution of the ratio of weak keys for different values of μ is listed in Table 5.1. The experimental results very closely follow the theoretical estimate.

μ	2	4	6	8	16
theoretical r_w	0.9988	0.9788	0.8843	0.6762	0.0220
experimental r_w	1	0.98	0.89	0.68	0.02

Table 5.1: Theoretical and experimental weak key ratio for SMALLPRESENT-24.

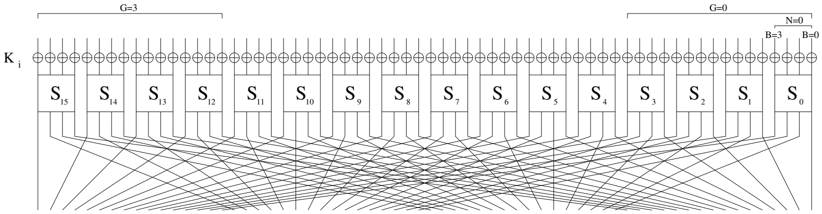


Figure 5.15: One round of the PRESENT block cipher.

5.7 Applications: Attacking PRESENT and Serpent

5.7.1 The block cipher PRESENT

The block cipher PRESENT is designed by Bogdanov *et al.* as a very lightweight cipher [32]. It has a 31-round SPN structure in which the S-box layer has 16 parallel 4-bit S-boxes and the diffusion layer is a bit permutation [32]. The block size is 64 bits and the key size can be 80 bits or 128 bits. One round of PRESENT is illustrated in Figure 5.15.

PRESENT is one of the few significant block cipher proposals after the standardisation of the AES, and has recently been standardised by ISO. As such, it has been extensively analysed. Wang presented a differential attack on 16-round PRESENT [160]. Collard *et al.* proposed a new cryptanalytic technique, the statistical saturation attack, to attack 24-round PRESENT [48]. There are multiple papers presenting attacks based on linear hulls for PRESENT [45, 128, 139], leading to linear attacks for up to 26 rounds [45], which is also the currently best known cryptanalytic result on PRESENT. Since the S-box of PRESENT admits linear approximations with single-bit linear masks, the attacker can exploit linear hulls containing many single-bit linear trails over an arbitrary number of rounds [139]. However, this property does not hold in the case of differential cryptanalysis, so for differential attacks, paths in which two active S-boxes appear per round have to be used. Hence, a linear attack on PRESENT will typically be more efficient than differential attacks.

Blondeau and Gérard use multiple differentials to attack 18 rounds of the PRESENT block cipher [29]. The attack as presented in [29] however suffers from a technical inaccuracy, invalidating the attack [28, 161]. Using different sets of differentials, this has been corrected in [27]. However, as outlined in [161], also this attack needs a minor correction which decreases its efficiency.

5.7.2 Identifying good differential paths for PRESENT

The diffusion layer of PRESENT partitions the 64 state bits in four *groups* G_0, G_1, G_2, G_3 with G_i encompassing the state bits $16i, \dots, 16i+15$. Due to the properties of PRESENT's S-box, we focus on finding differential characteristics with two active S-boxes in each round. The foundation for this search is formulated in Theorem 5.19. Based on this theorem, an efficient search algorithm for differential characteristics with two active S-boxes per round can be built.

Theorem 5.19. *For the PRESENT block cipher, differential characteristics with only two active S-boxes per round must have the following pattern:*

1. *If two active S-boxes are in the same group in round r, their output difference will be equal and must have two non-zero bits to ensure that only two active S-boxes appear in the $(r+2)$-nd round, and the two active S-boxes in round $r+1$ will be in the different groups;*
2. *If two active S-boxes are in different groups in round r, their output difference will be equal and must have only one non-zero bit to ensure that only two active S-boxes appear in the $(r+1)$-st round, and the two active S-boxes in round $r+1$ will be in the same group.*

For a proof of Theorem 3, we refer to [161].

Based on this observation, 91 16-round differentials with only two active S-boxes in each round and a single output difference $\Delta_{16} = 00000500_x || 00000500_x$ with probabilities ranging from $2^{-62.13}$ to $2^{-63.95}$ have been identified. The full list of these differentials can be found in [161].

5.7.3 Key recovery attack on 18-round PRESENT-80

In this section, we briefly describe how to use the 16-round differentials identified in the last section to attack 18-round PRESENT with an 80-bit key. The full attack procedure is detailed in [161].

The first step is to choose the set of differentials. From the output difference $00000500_x||00000500_x$ at round 16, we can derive that the number of recovered subkey bits in round 17 and round 18 is $8 + 32 = 40$. Those 40 subkey bits are independent according to the key schedule. In this attack, we will use the whole codebook and set the size of the candidates of subkey counters ℓ to 2^{36}. With Equation (5.21), we have $n_k = 40$, $\ell = 2^{36}$ and $N = 2^{64}$, and the success probability is maximised at 85.95% as we take the 36 16-round differentials with the highest probabilities of the 91 differentials found by the search algorithm. Therefore, $|\Delta_0| = 36$. Note that this is an example that it is necessary to carefully evaluate the impact of choosing the set of differentials. Simply adding more input differences to further decrease the data complexity does not necessarily minimise the overall complexities.

We construct 2^{24} structures of 2^{40} chosen plaintexts each. In each structure, all the inputs to the 6 non-active S-boxes in the first round take a fixed random value, while 40 bits of input to 10 active S-boxes take 2^{40} possible values. In all structures, there are $2^{24} \cdot 2^{39} = 2^{63}$ pairs for each possible differential. The sum of the probabilities for all 36 differentials is $2^{-57.97}$, so the number of right pairs is $2^{63} \cdot 2^{-57.97} = 2^{5.03}$. The filter probability for the ciphertext pairs β according to active S-boxes can be estimated at $\beta = 2^{-12.55}$ [161].

In the terminology of Section 5.6, we have $|\Delta_0| = 36$, $\sum_{i=1}^{|\Delta_0|} p_i = 2^{-57.97}$, $N_{st} = 24$, $N_p = 40$, $N_c = 32$, $n_k = 40$ and $\ell = 2^{36}$.

According to the model of Section 5.6.2, the total time complexity will be 2^{76} 18-round encryptions. The data complexity is 2^{64} chosen plaintexts and the memory requirements are 2^{40} 128-bit cells for the hash table, which can be reused for the 2^{40} counters. The success probability is 85.95%.

The ratio of weak key satisfying the sum of the probabilities of the 36 differentials is computed as follows:

$$r_w = 1 - \sum_{x=0}^{\mu-1} \text{Poisson}(x, 2^{n-1} \sum_{j=1}^{|\Delta_0|} p_i) = 1 - \sum_{x=0}^{2^{5.03}-1} \text{Poisson}(x, 2^{63} \cdot 2^{-57.97}) = 0.57.$$

This means that the number of weak keys for which our attack can succeed is $2^{80} \cdot 0.57 = 2^{79.19}$ for PRESENT-80.

Comparison to previous attacks on 18-round PRESENT

As outlined in Section 5.7.1, the multiple differential attack of [29] does not work as described. It has been superseded by [27], where another multiple differential attack on 18-round PRESENT is presented. It can be seen from

Attack of [27]		**Attack of Section 5.7.3**			
ℓ	P_S	ℓ	P_S	N	time complexity
2^{38}	65.27%	$\mathbf{2^{36}}$	**85.94%**	2^{64}	2^{76}
2^{39}	79.68%	$\mathbf{2^{37}}$	**92.30%**	2^{64}	2^{77}
2^{41}	94.62%	$\mathbf{2^{39}}$	**98.36%**	2^{64}	2^{79}

Table 5.2: Comparison of our attacks on PRESENT with the multiple differential cryptanalysis of [27].

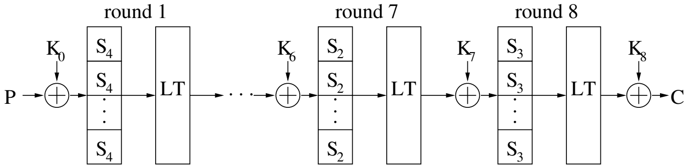


Figure 5.16: The block cipher Serpent reduced to 8 rounds.

Table 4 of [27], that $|\Delta_0| = 17$ (and not 16 as assumed in the paper). This results in $p_* = 2^{-62.6765}$ (instead of $2^{-62.59}$) and $p = 2^{-63.56}$ (instead of p=$2^{63.47}$). Based on these values, we compare this attack to our attack from Section 5.7.3 for different values of the number ℓ of remaining key candidates (see Table 5.2). One can see that for the same data and time complexities, the structure attack performs consistently better than multiple differential cryptanalysis with multiple input differences and multiple output differences.

5.7.4 Attack on reduced-round Serpent

Serpent was one of the five AES candidates in the final round; it is a substitution-permutation network 4-bit S-boxes, a linear diffusion layer and 32 rounds in total. For a full specification, we refer to [3]. Figure 5.16 depicts Serpent reduced to 8 rounds, from round 4 to 11.

The best known cryptanalytic result is the differential-linear cryptanalysis on 12 rounds [68]. In [20], Biham *et al.* describe a differential attack for 7-round Serpent with a data complexity of 2^{84} chosen plaintexts and a time complexity of 2^{85} memory accesses. Biham *et al.* also give a differential attack on 8-round Serpent-256 with 2^{213} memory accesses and 2^{84} chosen plaintexts. Those are the best known *differential* attacks on Serpent.

In [20], 2^{14} differential characteristics for $\frac{1}{2} + 5$ rounds are constructed, starting from the linear transformation with fewer active S-boxes (13 active S-boxes) in the first half round, then extending them backwards to 6 rounds. All differential characteristics have an output difference of $\{0906b010_x||00000080_x||13000226_x||06040030_x\}$.

We now apply the structure attack to Serpent. We will use the same output difference as [20], but since all differentials have high probability compared to 2^{-128}, we will make use of all the possible non-zero input differences according to the output differences for the S-boxes in the first round. This leads to $|\Delta_0| = 2^{35.32}$ and $\sum_{i=1}^{|\Delta_0|} p_i = 2^{-65}$.

We construct 2^{19} structures of 2^{52} chosen plaintexts each. In each structure, all the inputs to non-active S-boxes in the first round are fixed to some random value, while the 52 bits of input to all the active S-boxes take all the 2^{52} possible values. There are $2^{19} \cdot 2^{51} = 2^{70}$ pairs for each differential characteristic. We expect that about $2^{70} \cdot 2^{-65} = 2^5$ pairs produce the output difference Δ_6.

The success probability P_S can be computed with Equation (5.21). Here $N = 2^{71}$, $|\Delta_0| = 2^{35.32}$, $p_* = 2^{-65} \cdot 2^{-35.32} \cdot 2^{52} = 2^{-48.32}$, $N_s = 2^{70} \cdot 2^{35.32} \cdot 2^{-52} = 2^{53.32}$, $p = 2^{-52}$, $n_k = 52$, $l = 2$, $\beta = 2^{-26.22}$, hence we get $P_S = 89.87\%$.

The time complexity is $2^{27.10} \cdot 2^{52} \cdot 13/32 = 2^{77.81}$ one-round encryptions which is equivalent to $2^{74.99}$ 7-round encryptions, the data complexity is 2^{71} chosen plaintexts and the memory requirements are 2^{52} hash cells of 256 bits and 2^{52} 32-bit counters storing 2^5 pairs each, hence using about 2^{57} 256-bit words. This attack consequently applies to Serpent with all key sizes of 128,192 and 256 bits.

The attack can be further extended to 8-round Serpent-256. By exhaustively searching the 128-bit subkey in the last round to decrypt to round 7, the above attack for 7 rounds can be applied. The time complexity is $2^{203.81}$ 8-round encryptions, the data complexity is 2^{71} chosen plaintexts and the memory requirements are the same as for the 7-round attack. This attack therefore applies only to Serpent with a 256-bit key.

In comparison, the previous differential attack for 7-round Serpent described in [20] has a time complexity of 2^{85} and a data complexity of 2^{84} chosen plaintexts. For the previous differential attack on 8-round Serpent, the time complexity is 2^{213} and the data complexity is 2^{84} chosen plaintexts. This implies that our attacks require substantially less chosen plaintexts. This comparison is summarised in Table 5.3.

	Biham et al (2001)		**Structure attack**	
rounds	time	data	time	data
7	2^{85}	2^{84}	$\mathbf{2^{75}}$	$\mathbf{2^{71}}$
8	2^{213}	2^{84}	$\mathbf{2^{203}}$	$\mathbf{2^{71}}$

Table 5.3: Comparison of our structure attacks to previous differential attacks on Serpent.

The ratio of weak keys satisfying the probability of the multiple differentials is computed as follows:

$$r_w = 1 - \sum_{x=0}^{\mu-1} \text{Poisson}(x, 2^{n-1} \sum_{j=1}^{|\Delta_0|} p_i) = 1 - \sum_{x=0}^{2^5-1} \text{Poisson}(x, 2^{70} \cdot 2^{-65}) = 0.52.$$

This means that this attack is expected to work with about half of all possible keys, independent of the key size.

5.8 Conclusions

In this chapter, we studied two important statistical analysis techniques for block ciphers: Linear cryptanalysis using Matsui's Algorithm 2 and differential cryptanalysis using structures.

In both cases, we analysed how to model and estimate the data and time complexities of these attacks more accurately by applying the theory of the probability distributions of correlations of linear approximations and cardinality of differentials over Boolean permutations [60].

For linear cryptanalysis using Matsui's Algorithm 2, we first formulated a new wrong key randomisation hypothesis. Based on this result, we obtained a new and more accurate formula for the probability of success in a linear attack (Section 5.2). We demonstrated that this probability was previously underestimated when the attacker tries to exploit a linear approximation with low bias, to attain a high advantage over the brute force, or both. These cases are very typical since cryptanalysts always try to break as many rounds of the cipher as possible by pushing the attack to the limit.

As a surprising consequence of the adjusted wrong key randomisation hypothesis, our analysis revealed that the success probability in general is not a monotonous function of the data complexity. This means that sometimes, using less data can result in better success probability of a linear attack.

Furthermore, in Section 5.4, we studied the impact of the linear hull effect in the context of the widely assumed hypothesis of key equivalence. To address the conflict between these, we proposed a heuristic approach to split the linear hull into a signal part (sampled for random keys) an a statistically modeled noise part to make the complexity estimation more realistic.

All these theoretical observations and techniques have been verified by experiments with structurally different small-scale ciphers.

In the context of differential cryptanalysis, we gave a general model for the structure attack in Section 5.6, providing guidance on how to choose the set of differentials to minimize the overall complexities. As concrete applications of our model, we presented structure attacks on 18-round PRESENT and improved the previous differential cryptanalytic results for the Serpent block cipher. To the best of our knowledge, these attacks are the best known *differential* attacks on those two block ciphers.

Comparing our model for structure attacks against the general model for multiple differential cryptanalysis proposed in [29], we concluded that the limitation for the set of input differences imposed by the model of [29] excludes many valuable differentials. We show that in structure attacks, a very important – and often particularly efficient – subclass of multiple differential attacks, this restriction can be relaxed.

Chapter 6

Designing block ciphers: Statistical and structural aspects

In a certain sense, the question how to design a secure block cipher can be considered “solved”. Since the publication of the Data Encryption Standard DES [72], the cryptographic research community has developed a rich body of studies on the design and cryptanalysis of block ciphers. Most prominently, DES has provoked the discovery of differential [22] and linear cryptanalysis [115, 116], two of the most powerful attack techniques to date.

Soon after the publication of these attacks, study was initiated on the question how to demonstrate resistance against them [53,56,101,117,135,144,159]. This research culminated with the standardisation of the Advanced Encryption Standard (AES) by the National Institute of Standards and Technology (NIST). AES is designed according to the wide trail design strategy [56], which provides strong guarantees against the powerful statistical attacks. The works of Knudsen and Nyberg [101, 135], Daemen and Rijmen [56], Matsui [117] Vaudenay [159] all have this “practical security” approach in common: Demonstrate security against the most powerful known attacks (and some generalisations).

Being able to show that a particular block cipher resists known attacks is a great asset. However, the question naturally arises whether and to which extent generic resistance can be shown against attacks performing arbitrary computations only limited by the resources (time, data, storage) that the attacker has at their disposal. This approach commonly models the success (or: advantage over the random case) of an attacker via his ability to break a security notion based on having obtained a certain number of *queries* to the algorithm under attack. The computational abilities of the attacker are modeled either as a probabilistic polynomial time Turing machine (PPT) [79] (polynomial in the security parameter), or information-theoretically, meaning that the adversary is granted unlimited computational resources to process the limited number of queries he has obtained during the interaction. In this model, commonly referred to as "provable" or "reductionist security", the security of a scheme is not merely assessed with regard to specific attacks; however, the universality of the model requires the analysis to be based on making assumptions about the ideality of certain subcomponents of the algorithm.

Generally speaking, the practical security approach has led to a plethora of efficient and (as far as we know, practically secure) block cipher designs, while instantiating schemes for which reductionist security bounds can be proved, has proven difficult.

In this chapter, we mainly address two open issues in this context. First, the practical security approach heavily relies on the ability to preclude powerful statistical attacks. In order to convert this into design requirements, it is important to know a *reference point*, the ideal case that can be attained by any block cipher. Somewhat surprisingly, such a reference point has not yet been formulated for either differential or linear cryptanalysis. We propose such a reference point in Section 6.2.

Second, the concept of a key-alternating cipher is omnipresent in symmetric cryptography: Most block cipher proposals, including the AES, are key-alternating ciphers. The question whether this is a sound design strategy for block ciphers has not yet been studied theoretically, with the notable exception of the Even-Mansour construction [70] (which is a key-alternating cipher limited to only one round). This motivates a general study of the soundness of the concept of key-alternating ciphers, which is carried out in Sections 6.5 to 6.8.

6.1 Distributions of linear and differential properties

The underlying nonlinearity measures of linear and differential cryptanalysis are the correlation of linear approximations and cardinality of a differential, respectively. As outlined in Section 2.3, there are many equivalent ways of expressing these quantities. In this chapter, we will mostly deal with the discrete variants — imbalance of linear approximations and cardinality of differentials – in order to avoid ambiguities due to continuity correction. Note that statements about the derived quantities (correlation, Fourier coefficients, linear probability, linear bias, and differential probability) can readily be obtained by applying the relations mentioned in Sections 2.3.1 and 2.3.2.

6.1.1 Distributions over all functions and permutations

For vectorial Boolean functions, the following characterisations for the distribution of linear and differential properties have been proven.

Theorem 6.1 ([60, Theorem 4.2])**.** *The imbalance I_{α_0,β_0} of a linear approximation (α_0, β_0) for an n-bit to m-bit vectorial Boolean function has the following distribution, taken over all n-bit to m-bit vectorial Boolean functions:*

$$\Pr_f(I^f_{\alpha_0,\beta_0} = x) = 2^{-2n}\binom{2^n}{2^{n-1}+x}, \tag{6.1}$$

where x ranges from -2^{n-1} to 2^{n-1}.

For the case of permutations, the following result is known:

Theorem 6.2 ([137, Theorem 1], [60, Lemma 4.5])**.** *The imbalance I_{α_0,β_0} of a linear approximation (α_0, β_0) for an n-bit permutation has the following distribution, taken over all n-bit permutations:*

$$\Pr_\pi(I^\pi_{\alpha_0,\beta_0} = 2x) = \frac{\binom{2^{n-1}}{2^{n-2}+x}^2}{\binom{2^n}{2^{n-1}}}, \tag{6.2}$$

with x ranging from -2^{n-1} to 2^{n-1}.

The cardinality of differentials over vectorial Boolean functions is known to have a binomial distribution:

Theorem 6.3 ([60, Lemma 3.3])**.** *Over all n-bit to m-bit vectorial Boolean functions, the distribution of N_{α_0,β_0} for a fixed differential (α_0, β_0) follows a Binomial distribution:*

$$\Pr_f(N^f_{\alpha_0,\beta_0} = i) = (2^{-m})^i(1-2^{-m})^{2^{n-1}-i}\binom{2^{n-1}}{i}. \tag{6.3}$$

For permutations, this distribution has been characterised in [136, 138].

Theorem 6.4 ([138, Corollary 2.1]). *Over all n-bit permutations, the distribution of N_{α_0,β_0} for a fixed differential (α_0, β_0) is as follows:*

$$\Pr_{\pi \in S_{2^n}} (N^{\pi}_{\alpha_0,\beta_0} = k) = \binom{2^{n-1}}{k}^2 \cdot \frac{k! \cdot 2^k \cdot \Phi(2^{m-1} - k)}{2^n!} \tag{6.4}$$

with $\Phi(d) = \sum_{k=0}^{d} (-1)^k \cdot \binom{d}{k}^2 \cdot 2^k \cdot k! \cdot (2d - 2k)!$.

For sufficiently large n, those distributions can be closely approximated by either normal or Poisson distributions:

Theorem 6.5 ([60, Corollary 4.3, Lemma 4.6]). *When n is sufficiently large, the distributions of the imbalance for a fixed approximation (α_0, β_0) over all n-bit to m-bit Boolean functions given by (6.1) and over all n-bit permutations given by (6.2) can be approximated by continuous distributions as follows:*

$$\Pr_f (I^f_{\alpha_0,\beta_0} = z) \approx \mathcal{N}(0, 2^{n-2}), \tag{6.5}$$

$$\Pr_\pi (I^\pi_{\alpha_0,\beta_0} = 2z) \approx \mathcal{N}(0, 2^{n-4}). \tag{6.6}$$

Theorem 6.6 ([60, Corollary 3.6], [83, Lemma 5]). *For sufficiently large n, the distributions of cardinality of a fixed differential (α_0, β_0) over all n-bit to n-bit Boolean functions given by (6.3) and over all n-bit permutations given by (6.4) can be approximated by continuous distributions as follows:*

$$\Pr_g (N^g(\alpha_0, \beta_0) = z) \approx \frac{(1/2)^z e^{-1/2}}{z!} = \text{Poisson}(z, 1/2), \tag{6.7}$$

$$\Pr_\pi (N^\pi(\alpha_0, \beta_0) = z) \approx \text{Poisson}(z, 1/2). \tag{6.8}$$

6.2 Distributions for a randomly drawn function or permutation

6.2.1 Motivation

As seen in the results listed in the previous section, the distribution of the imbalance of linear approximations and cardinality of differentials for vectorial Boolean functions and permutations has previously been studied for *fixed* linear approximations or differentials, taking the probabilities over all Boolean functions or permutations of the appropriate dimensions.

In this section, we demonstrate how to obtain a statement over the distribution of these linear and differential properties for one randomly drawn instance (function or permutation), taking the probabilities over all linear approximations or differentials. It turns out that the previous analysis can carry over to this case.

This has important applications in symmetric-key cryptography, since these values are directly related to the complexity of linear and differential attacks. The value of correlation attained for a given linear approximation of a permutation or function determines the complexity of linear cryptanalysis. Similarly, the value of the cardinality for a certain differential characterises its probability and yields a complexity estimate of differential cryptanalysis.

The setting of randomly drawing a permutation and then studying the distribution of the nonlinearity measures over all parameters, corresponds to randomly choosing a key in an idealised block cipher or to randomly picking a compression function for a hash function. Consequently, characterising the distribution of correlation of linear approximations and cardinality of differentials for a fixed yet randomly chosen permutation or function defines *the reference point* for linear and differential cryptanalysis of block ciphers and hash functions.

In the following, let (α, β) be either a linear approximation or a differential. For each fixed value of (α, β), the results of Theorems 6.1–6.4 state the distribution of the proportion of functions resp. permutations with $I_{\alpha,\beta} = z$ or $N_{\alpha,\beta} = z$. We now argue that the distribution of those quantities with respect to one randomly drawn instance (function or permutation), where the probabilities are taken over all (α, β), can be derived from those known distributions.

6.2.2 On the expected nature of our results

It should be noted that the following analysis provides the distributions for a fixed but *randomly drawn* function or permutation, and not for any *fixed selected* instance. The distributions provided by the following proposition can be interpreted as the expected distributions when dealing with a fixed function or permutation which has been drawn uniformly at random from the set of all functions or permutations with the same dimensions.

6.2.3 Main result

The following proposition provides the sought statement about the distribution of imbalance of linear approximations and cardinality of differentials in a randomly drawn Boolean function or permutation.

Proposition 6.7. *Let f_0 be a Boolean function, drawn uniformly at random from the set of all n-bit to m-bit Boolean functions, and π_0 an n-bit permutation, drawn uniformly at random from the symmetric group S_{2^n}. Then we have*

$$\Pr_{(\alpha,\beta)}(I^{f_0}_{\alpha,\beta} = x) = 2^{-2n}\binom{2^n}{2^{n-1}+x}, \tag{6.9}$$

$$\Pr_{(\alpha,\beta)}(N^{f_0}_{\alpha,\beta} = i) = (2^{-m})^i(1-2^{-m})^{2^{n-1}-i}\binom{2^{n-1}}{i}, \tag{6.10}$$

and

$$\Pr_{(\alpha,\beta)}(I^{\pi_0}_{\alpha,\beta} = 2x) = \frac{\binom{2^{n-1}}{2^{n-2}+x}^2}{\binom{2^n}{2^{n-1}}}, \tag{6.11}$$

$$\Pr_{(\alpha,\beta)}(N^{\pi_0}_{\alpha,\beta} = k) = \binom{2^{n-1}}{k}^2 \cdot \frac{k! \cdot 2^k \cdot \Phi(2^{m-1}-k)}{2^n!} \tag{6.12}$$

with $\Phi(d) = \sum_{k=0}^{d}(-1)^k \cdot \binom{d}{k}^2 \cdot 2^k \cdot k! \cdot (2d-2k)!$.

Proof. Let $Q_{\alpha,\beta}$ denote either the imbalance $I_{\alpha,\beta}$ of an approximation (α,β) or the cardinality $N_{\alpha,\beta}$ of the differential (α,β). Denote by $\mathcal{F}$ either the set of all n to m-bit vectorial Boolean functions or n-bit permutations. We will prove the statements (6.9)–(6.12) by showing that for all z, the probability that $Q_{\alpha,\beta} = z$, taken over all (α,β) for one $g_0 \in \mathcal{F}$ drawn uniformly and at random, equals the probability that $Q_{\alpha,\beta} = z$ for an arbitrary but fixed (α,β), taken over all $f \in \mathcal{F}$:

$$\Pr_{f\in\mathcal{F}}(Q_{\alpha,\beta} = z) = \Pr_{(\alpha,\beta)}(Q^{g_0}_{\alpha,\beta} = z).$$

The claims then follow immediately from Theorems 6.1–6.4.

From Theorems 6.1–6.4, we know the probabilities

$$\Pr_{f\in\mathcal{F}}(Q_{\alpha,\beta} = z) = \frac{\#\{f \mid Q^f_{\alpha,\beta} = z\}}{\#\mathcal{F}}$$

for fixed α,β. Now, we draw one $g_0 \in \mathcal{F}$ uniformly at random and want to determine the quantity

$$\frac{\#\{(\alpha,\beta) \mid Q^{g_0}_{\alpha,\beta} = z_0\}}{\#\{(\alpha,\beta)\}} = \Pr_{(\alpha,\beta)}(Q^{g_0}_{\alpha,\beta} = z_0)$$

for each z_0. By the law of total probability, we can write

$$\begin{aligned}
\Pr_{(\alpha,\beta)}(Q^{g_0}_{\alpha,\beta} = z_0) &= \sum_{(\alpha,\beta)} \Pr((\alpha,\beta)) \Pr\left(Q^{g_0}_{\alpha,\beta} = z_0 \mid (\alpha,\beta)\right) \\
&\overset{(1)}{=} \sum_{(\alpha,\beta)} \Pr((\alpha,\beta)) \frac{\#\{f \mid Q^f_{\alpha,\beta} = z_0\}}{\#\mathcal{F}} \\
&\overset{(2)}{=} \frac{\#\{f \mid Q^f_{\alpha,\beta} = z_0\}}{\#\mathcal{F}} \underbrace{\sum_{(\alpha,\beta)} \Pr((\alpha,\beta))}_{=1} \\
&= \frac{\#\{f \mid Q^f_{\alpha,\beta} = z_0\}}{\#\mathcal{F}} \\
&= \Pr_{f\in\mathcal{F}}(Q^f_{\alpha,\beta} = z_0).
\end{aligned}$$

For step (1), we use the fact that g_0 is drawn uniformly at random from $\mathcal{F}$: For fixed (α,β) and z_0, the probability that $Q_{\alpha,\beta} = z_0$ given (α,β) is equal to the probability that g_0 is equal to one of the elements of $\mathcal{F}$ for which (α,β) yields $Q_{\alpha,\beta} = z_0$, which is in turn equal to the proportion of elements in $\mathcal{F}$ for which $Q_{\alpha,\beta} = z_0$ for this fixed (α,β).

For step (2), we use that the distributions of $\Pr_{f\in\mathcal{F}}(Q_{\alpha,\beta} = z)$, and therefore the quantities $\#\{f \mid Q^f_{\alpha,\beta} = z\}$, are identical for all (α,β).

We can also directly see that this forms a valid probability distribution: We know that for all fixed (α,β), $\sum_z \Pr_f(Q_{\alpha,\beta} = z) = 1$, so consequently for one randomly drawn g_0, $\sum_z \Pr_{(\alpha,\beta)}(Q^{g_0}_{\alpha,\beta} = z) = 1$ as well. ☐

Note also that the above derivation states that for each z_0, $\Pr_{(\alpha,\beta)}(Q_{\alpha,\beta} = z_0)$ and therefore $\Pr_{f\in\mathcal{F}}(Q_{\alpha,\beta} = z_0)$ gives the expected value of the conditional probability that $Q_{\alpha,\beta} = z_0$ given (α,β):

$$\Pr_{(\alpha,\beta)}(Q_{\alpha,\beta} = z_0) = \mathbf{E}_{(\alpha,\beta)}\left[\Pr\left(Q_{\alpha,\beta} = z_0 \mid (\alpha,\beta)\right)\right].$$

We stress that for the statement of this proposition to hold, $\mathcal{F}$ is required to be the complete set of either all n-bit to m-bit Boolean functions or all n-bit permutations. Distributions over subsets will have related, but different distributions. Similarly, our results hold for an instance g_0 that has been drawn uniformly at random from $\mathcal{F}$.

As in the case of distributions for fixed (α_0, β_0) over all functions or permutations, the distributions of Proposition 6.7 can be closely approximated by continuous distributions:

Corollary 6.8. *Let f_0 a randomly drawn n-bit to m-bit Boolean function, g_0 a randomly drawn n-bit Boolean transformation, and π_0 a randomly drawn n-bit permutation. For sufficiently large n, the distributions of the imbalance over all approximations (α, β) and the distributions of cardinality over all differentials (α, β) can be approximated by continuous distributions:*

$$\Pr_{(\alpha,\beta)}(I^{f_0}_{\alpha,\beta} = z) \approx \mathcal{N}(0, 2^{n-2}), \tag{6.13}$$

$$\Pr_{(\alpha,\beta)}(I^{\pi_0}_{\alpha,\beta} = 2z) \approx \mathcal{N}(0, 2^{n-4}), \tag{6.14}$$

$$\Pr_{(\alpha,\beta)}(N^{g_0}(\alpha, \beta) = z) \approx \mathrm{Poisson}(z, 1/2), \tag{6.15}$$

$$\Pr_{(\alpha,\beta)}(N^{\pi_0}(\alpha, \beta) = z) \approx \mathrm{Poisson}(z, 1/2). \tag{6.16}$$

6.2.4 Discussion

The results of Section 6.1.1 provide a way of characterising the best conceivable behaviour with regard to linear and differential cryptanalysis. In a word, they define a *reference point* that a cryptographically good block cipher (for any random choice of its secret key) or a cryptographically good permutation or compression function for construction of a cryptographic hash function should aim to attain in terms of resistance to differential and linear cryptanalysis.

An concrete example in the case of hash function design will be provided in Section 6.2.6.

For the case of block cipher design, consider the situation when one wants to build a cipher from a fixed set of underlying public (non-keyed) permutations. One strategy to select this set of permutations is by selecting some keys in a secure block cipher. The expected distributions of correlation and cardinality of differentials for these permutations are then given by the results in this section. If the designer can argue that his choice conforms with these distributions, this constitutes a strong point for the design.

A further consequence of the results listed in Section 6.1.1 is that if an attacker fixes any linear approximation or differential, there are very few functions or permutations for which those yield good results. Dually, it follows from

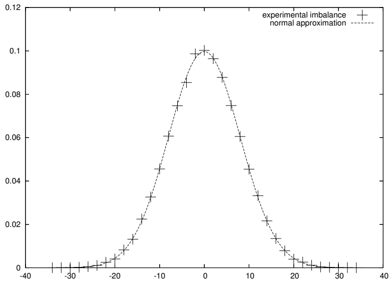


Figure 6.1: Experimental distribution of the imbalance for a single randomly drawn permutation and its normal approximation.

Proposition 6.7 that in a randomly drawn function or permutation, there are very few useful approximations or differentials.

6.2.5 Numerical illustration

We now illustrate the theoretical result of Proposition 6.7 by means of numerical experiments. Firstly, a single 8-bit permutation was drawn uniformly at random from the set of all 8-bit permutations, and the distributions of the imbalances of all nontrivial linear approximations and the cardinality of all nontrivial differentials were compiled. This experiment has been repeated several times, always yielding stable results. The experimental and theoretical distributions are depicted in Figures 6.1 and 6.2. It turns out that already in the case of a single permutation, the experimentally observed distributions match the continuous approximations to the theoretical distributions extremely closely.

In a second experiment, the distributions of the average imbalance and cardinality over 2000 randomly drawn permutations have been considered. Besides improving the accuracy for the extreme tails, no significant difference to the case of considering a single randomly drawn permutation was observed.

6.2.6 On related results

Even though to the best of our knowledge, the distributions of Proposition 6.7 have not previously been studied in their full generality, there are two important related studies by Daemen *et al.* [15] and Daemen and Rijmen [60], which we describe in the following.

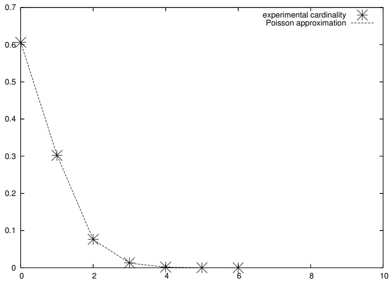


Figure 6.2: Experimental distribution of the cardinality of differentials for a single randomly drawn permutation and its Poisson approximation.

Experimental analysis of the Keccak permutation reduced to 25 bits

The Keccak hash function is a design based on the Sponge construction and currently a final round candidate for selection as SHA-3 standard by NIST [17]. It is based on a iterated permutation Keccak-f. After a sufficient number of rounds, Keccak-f is expected to behave as a randomly drawn permutation.

The designers of the Keccak hash function have experimentally analysed these distributions for the underlying permutation of their design. More precisely, the distribution of correlation of linear approximations and cardinality of differentials of the Keccak-f permutation reduced to 25 bits (denoted Keccak-f[25]) have been analysed experimentally [17, Sect. 4.3.2 and 4.3.3]. Based on sampling 2^{41} of all $(2^{25}-1)^2$ nontrivial differentials and 2^{39} of all $(2^{25}-1)^2$ nontrivial linear approximations, these experiments demonstrate that the distributions for a single “random-looking” 25-bit permutation match the theoretical distributions of a single differential or approximation over all 25-bit permutations very closely.

Proposition 6.7 now explains this behaviour also theoretically. Conversely, the experiments on Keccak-f[25] provide further support for the validity our results, especially the continuous approximations of Corollary 6.8.

As described in Section 6.2.4, the results of Proposition 6.7 provide a theoretical foundation for the experimental claim that the Keccak-f permutation behaves close to the appropriate reference point (a randomly drawn permutation for building a hash function) with regard to statistical attacks.

Distribution of maxima

In [60, Sect. 5], the distributions of maximum cardinality $\max_{\alpha,\beta} N_{\alpha,\beta}$ and linear probability $\max_{\alpha,\beta} \mathrm{LP}_{\alpha,\beta}$ over a large set of n-bit permutations are shown to very closely resemble log-Weibull distributions with small standard deviation. In particular, this implies that the distributions of the maxima are very narrowly concentrated at two or three successive integer values (in the case of cardinality) or multiples of 2^{-n} (in the case of LP).

By comparison, our results state the shape of the complete distributions of the imbalances and cardinalities for a single randomly drawn permutation over all (α, β). Additionally, our experimental results closely match the experimental data from [60]: As seen in Figure 6.1, the maximum cardinality observed for a randomly drawn 8-bit permutation was $N = 6$, which coincides with the peak value of the extreme value distribution in Fig. 2 of [60]. Similarly, the maximum observed LP for one randomly drawn 8-bit permutation was $\frac{18}{256}$, which again closely matches the peak value in Figure 3 of [60].

6.3 High-probability approximations and differentials for a permutation

Linear and differential cryptanalysis exploit linear approximations with a high correlation and differentials with large cardinalities. Recall that the complexity of a linear attack using a linear approximation (α, β) is roughly proportional to $|C_{\alpha,\beta}|^{-2}$, while the complexity of a differential attack using a differential (α, β) is roughly proportional to $\mathrm{DP}_{\alpha,\beta}^{-1}$. Using Proposition 6.7, we can now explicitly determine the probabilities that a randomly drawn permutation has linear approximations or differentials with high correlation resp. cardinality.

Proposition 6.9. *For a randomly drawn permutation* π_0 *and an integer* $x \geq 0$*, the probability of a correlation with absolute value higher than* $2^{-n/2+x}$ *is*

$$\Pr_{(\alpha,\beta)}\left(|C^{\pi_0}_{\alpha,\beta}| \geq 2^{-n/2+x}\right) \approx \operatorname{erfc}\left(2^{x-1/2}\right).$$

Proof. We use the continuous normal approximation $\mathcal{N}(0, 2^{-n})$ to the discrete distribution (6.11). Since the distribution is symmetric around zero, we have

$$\begin{aligned}\Pr_{(\alpha,\beta)}(|C^{\pi_0}_{\alpha,\beta}| \geq 2^{-n/2+x}) &= 2\Pr_{\alpha,\beta}(C^{\pi_0}_{\alpha,\beta} \geq 2^{n/2+x})\\ &= 2\int_{2^{n/2+x}}^{2^{n-2}} \frac{1}{\sqrt{2\pi}2^{\frac{n-4}{2}}} e^{\frac{-z^2}{2^{n-3}}} dz,\end{aligned}$$

see Proposition 2 of [35]. Since $\Pr(C_{\alpha,\beta} = z) = 0$ for $z > 2^{n-2}$, we can write this as

$$\begin{aligned}&= \frac{2}{\sqrt{2\pi}2^{\frac{n-4}{2}}} \int_{2^{n/2+x}}^{\infty} e^{\frac{-z^2}{2^{n-3}}} dz\\ &= \frac{2}{\sqrt{2\pi}2^{\frac{n-4}{2}}} \lim_{a\to\infty} \left[\frac{\sqrt{\pi}\sqrt{2^{n-3}}}{2} \operatorname{erf}\left(\frac{z}{\sqrt{2^{n-3}}}\right)\right]^{a}_{z=2^{n/2+x}}\\ &= \frac{2}{\sqrt{2\pi}2^{\frac{n-4}{2}}} \cdot \frac{\sqrt{\pi}\sqrt{2^{n-3}}}{2} \lim_{a\to\infty} \underbrace{\operatorname{erf}\left(\frac{a}{\sqrt{2^{n-3}}}\right)}_{\to 1} - \operatorname{erf}\left(\frac{2^{n/2-2+x}}{\sqrt{2^{n-3}}}\right)\\ &= \frac{\sqrt{2\cdot 2^{n-4}}}{\sqrt{2}\cdot 2^{(n-4)/2}} \cdot \left(1 - \operatorname{erf}\left(2^{n/2-2+x-(n-3)/2}\right)\right)\\ &= 1 \cdot \operatorname{erfc}\left(2^{x-1/2}\right). \qquad \square\end{aligned}$$

For the differential case, we first prove a technical result:

Lemma 6.10. *For any integer $n \geq 0$ and $x \in \mathbb{R}$,*

$$\Gamma(2^n, x) = (2^n - 1)! \cdot e^{-x} \sum_{i=0}^{2^n-1} \frac{x^i}{i!}. \tag{6.17}$$

Proof. We prove the statement by induction. By definition of $\Gamma(n, x)$,

$$\Gamma(2^n, x) = \int_x^{\infty} e^{-t} t^{2^n-1} dt.$$

For $n = 0$, this yields

$$\Gamma(1, x) = \int_x^{\infty} e^{-t} t^{1-1} dt = -e^{-t}\Big|_{t=x}^{\infty} = e^{-x} = 0! \cdot e^{-x} \sum_{i=0}^{0} \frac{x^i}{i!},$$

so the statement is true for $n = 0$. Assume now that (6.17) holds for some $n \geq 0$. Integrating by parts, we have

$$\begin{aligned}\Gamma\big(2^{n+1}, x\big) = \int_x^\infty e^{-t} t^{2^n} dt &= \Big[-e^{-t} t^{2^n - 1}\Big]_{t=x}^{\infty} - \int_x^\infty -e^{-t} 2^n \cdot t^{2^n-1} dt \\ &= e^{-x} x^{2^n} + 2^n \int_x^\infty e^{-t} t^{2^n - 1} dt\end{aligned}$$

and using the induction hypothesis

$$\begin{aligned} &= e^{-x} x^{2^n} + 2^n \left((2^n - 1)! \cdot e^{-x} \sum_{i=0}^{2^n-1} \frac{x^i}{i!} \right) \\ &= 2^n! \cdot e^{-x} \cdot \left(\sum_{i=0}^{2^n-1} \frac{x^i}{i!} + \frac{x^{2^n}}{2^n!} \right) \\ &= 2^n! \cdot e^{-x} \sum_{i=0}^{2^n} \frac{x^i}{i!}, \end{aligned}$$

as claimed. □

Based on this lemma, we can derive

Proposition 6.11. *Let $x \geq 0$ be an integer. For a randomly drawn permutation π_0, the probability of a differential with cardinality greater or equal to 2^x is*

$$\Pr_{(\alpha,\beta)}\big(N^{\pi_0}_{\alpha,\beta} \geq 2^x\big) \approx 1 - \frac{\Gamma\big(2^x, \frac{1}{2}\big)}{(2^x - 1)!}.$$

Proof. Using the Poisson approximation for the binomial distribution (6.12), we have $\Pr_{(\alpha,\beta)}(N^{\pi_0}_{\alpha,\beta} = k) \approx \frac{e^{-1/2}}{2^k \cdot k!}$ and therefore

$$\begin{aligned}\Pr_{(\alpha,\beta)}\big(N^{\pi_0}_{\alpha,\beta} \geq 2^x\big) &\approx \sum_{z=2^x}^{\infty} \frac{e^{-1/2}}{2^z \cdot z!} \\ &= \underbrace{\sum_{z=0}^{\infty} \frac{e^{-1/2}}{2^z \cdot z!}}_{=1} - \sum_{z=0}^{2^x-1} \frac{e^{-1/2}}{2^z \cdot z!} \\ &= 1 - \frac{(2^x - 1)! \cdot e^{-1/2} \sum_{z=0}^{2^x-1} \frac{(1/2)^z}{z!}}{(2^x - 1)!}\end{aligned}$$

Table 6.1: Numerical illustration of propositions 6.9 and 6.11. (α, β) denotes either a linear approximation or a differential for the randomly drawn n-bit permutation π_0.

Correlation c	$\Pr_{(\alpha,\beta)}(\lvert C^{\pi_0}_{\alpha,\beta}\rvert \geq c)$	DP d	$\Pr_{(\alpha,\beta)}(\mathrm{DP}^{\pi_0}_{\alpha,\beta} \geq d)$
$2^{-n/2}$	0.317	2^{-n}	0.393
$2^{(1-n)/2}$	$2^{-2.67}$	2^{1-n}	$2^{-3.47}$
$2^{(2-n)/2}$	$2^{-4.46}$	2^{2-n}	$2^{-9.16}$
$2^{(3-n)/2}$	$2^{-7.74}$	2^{3-n}	$2^{-23.94}$
$2^{(4-n)/2}$	$2^{-13.95}$	2^{4-n}	$2^{-60.93}$

which can be reformulated by Lemma 6.10 as

$$= 1 - \frac{\Gamma\left(2^x, \frac{1}{2}\right)}{(2^x - 1)!}$$

□

Note that in both cases, probabilities do *not* depend on the dimension n, but only on x, which determines the fraction of the tail of the distribution we are interested in.

Propositions 6.9 and 6.11 are numerically illustrated in Table 6.1. The table lists the corresponding probabilities starting from the minimal values of the correlation or differential probability that are usable in an attack, namely $2^{-n/2}$ for a linear and 2^{-n} for a differential attack. The values given in one row correspond to approximately the same attack complexity.

By Proposition 6.7, those results immediately carry over to the tails of the distributions of correlation and cardinality of differentials for a fixed linear approximation or differential, taken over all n-bit permutations:

Corollary 6.12. *Consider a fixed linear approximation (α_0, β_0). Over all n-bit permutations, the probability of a correlation C_{α_0,β_0} with absolute value greater or equal to $2^{-n/2+x}$ is*

$$\Pr_{\pi}(\lvert C^{\pi}_{\alpha_0,\beta_0}\rvert \geq 2^{-n/2+x}) \approx \operatorname{erfc}\left(2^{x-1/2}\right). \tag{6.18}$$

Corollary 6.13. *Consider a fixed differential (α_0, β_0). Over all n-bit permutations, the probability of a cardinality N_{α_0,β_0} of (α_0, β_0) of at least 2^x is*

$$\Pr_{\pi}(N^{\pi}_{\alpha_0,\beta_0} \geq 2^x) \approx 1 - \frac{\Gamma\left(2^x, \frac{1}{2}\right)}{(2^x - 1)!}. \tag{6.19}$$

Consequently, the values given in Table 6.1 also apply to the probabilities of both corollaries.

The results of this section can be interpreted as follows. In the form of Propositions 6.9 and 6.11, they are relevant from a designer's point of view: Even if a block cipher, compression function or underlying permutation satisfies the reference points given by Proposition 6.7, high-probability differential and linear properties will exist to the extent described here by Propositions 6.9 and 6.11.

Dually, in the form of Corollaries 6.12 and 6.13, they are relevant from the cryptanalysts' point of view: After picking a certain linear approximation of differential in an attack, testing this against wrong guesses for the key will result in false alarms from about this fraction of all n-bit permutations. The impact of this in the success of a linear attack has been demonstrated in Section 5.2.

6.4 Impossible differentials and zero correlation approximations for a permutation

Impossible differential [19] and zero-correlation cryptanalysis [35] are two extensions of differential and linear cryptanalysis exploiting the impossibility of certain differentials or linear approximations, in contrast to conventional differential and linear attacks, which exploit differentials and linear approximations that have high probability.

The results of Proposition 6.7 immediately allow us to explicitly characterise the probability that an impossible differential or a zero correlation approximation occurs in a randomly drawn Boolean function or permutation. Analogously to the case of conventional differential and linear cryptanalysis, this formulates a *reference point* for the resistance against these extended attacks.

Proposition 6.14. *Let f_0 be a randomly drawn n-bit to m-bit Boolean function and π_0 a randomly drawn n-bit permutation. The probability of f_0 or π_0 to have a linear approximation with zero correlation is approximately given by*

$$\Pr_{\alpha,\beta}(C^{f_0}_{\alpha,\beta} = 0) \approx \frac{1}{\sqrt{2\pi}} 2^{\frac{2-n}{2}} \tag{6.20}$$

$$\text{and } \Pr_{\alpha,\beta}(C^{\pi_0}_{\alpha,\beta} = 0) \approx \frac{1}{\sqrt{2\pi}} 2^{\frac{4-n}{2}}, \tag{6.21}$$

respectively.

Table 6.2: Probability of a zero correlation approximation in a randomly drawn n-bit permutation π_0 for common values of n.

Permutation size n (bits)	8	32	64	128
$\Pr_{\alpha,\beta}(C^{\pi_0}_{\alpha,\beta} = 0)$	$2^{-3.33}$	$2^{-15.33}$	$2^{-31.33}$	$2^{-63.33}$

Proof. Considering distributions for a fixed linear approximation (α, β), this has been shown for permutations in Proposition 2 of [35]. The result for functions follows analogously from Corollary 4.3 in [60]. In both cases, invocation of Proposition 6.7 then establishes the claim for distributions over all (α, β) for randomly drawn instances. □

The probability of zero correlation for a randomly drawn permutation is numerically illustrated in Table 6.2 for some common permutation sizes.

A similar statement can be made about impossible differentials. It should be noted that the probability of an impossible differential to occur in a randomly drawn function or permutation has to be greater or equal to one half, which is significantly higher than in the case of zero correlation approximations:

Proposition 6.15 (Folklore)**.** *At least half of all differentials of an n-bit Boolean transformation or permutation are impossible.*

Proof. It is known that for all α, $\sum_{\beta \in \mathbb{F}_2^n} N(\alpha, \beta) = 2^{n-1}$, and since there are 2^n summands, the fraction of differentials with cardinality zero cannot be less than $2^{n-1}/2^n = 1/2$. □

Proposition 6.16. *Let f_0 be a randomly drawn n-bit to n-bit Boolean transformation and π_0 a randomly drawn n-bit permutation. The probability of f_0 or π_0 to have an impossible differential (i.e., a differential with cardinality zero) is approximately given by*

$$\Pr_{\alpha,\beta}(N^{f_0}_{\alpha,\beta} = 0) \approx e^{-1/2} \tag{6.22}$$

$$\text{and } \Pr_{\alpha,\beta}(N^{\pi_0}_{\alpha,\beta} = 0) \approx e^{-1/2}. \tag{6.23}$$

Proof. This follows directly from Corollary 3.6 in [60] and an invocation of Proposition 6.7. □

For a randomly drawn n-bit to m-bit Boolean function f_0, the probability to have an impossible differential can be calculated in an exact way by means of Lemma 3.3 of [60]:

$$\Pr_{\alpha,\beta}(N^{f_0}_{\alpha,\beta} = 0) = \left(1 - 2^{-m}\right)^{2^{n-1}} .$$

For a Boolean transformation (i.e., $n = m$) of sufficiently large dimension, this probability is then equal to

$$\left(1 - 2^{-n}\right)^{2^{n-1}} = \left(1 - \frac{2^{-1}}{2^{n-1}}\right)^{2^{n-1}} ,$$

which rapidly converges to $e^{-1/2} \approx 0.6065$ as n tends to infinity.

6.5 Structural properties of key-alternating ciphers

The remainder of this chapter is dedicated to the study of key-alternating ciphers. Most block cipher proposals, including the AES, are key-alternating ciphers [58]. However, the question whether this is a sound design strategy for block ciphers has not yet been studied theoretically in full.

A natural setting to evaluate the soundness of key-alternating ciphers with t rounds is to study whether it constitutes a secure way of constructing an n-bit block cipher from t public unkeyed random permutations. We can consider the key-alternating sound if it cannot be generically attacked with random (=ideal) components, or if the cryptographically relevant statistical properties of random permutations do carry over to the overall construction.

By modeling the round functions as public random permutations, we establish an important difference of our setting to that of an idealised Feistel cipher, the Luby-Rackoff construction [113]. For the Luby-Rackoff construction, it is necessary that for each key, the F-function used in the Feistel network is chosen afresh at random. This not only provides an immediate obstacle to practical implementability, but also constitutes a stronger theoretical assumption for achieving security.

6.5.1 Previous work

The key-alternating construction has been studied theoretically for the simplest case of just one round by Even and Mansour [69, 70]. The Even-Mansour

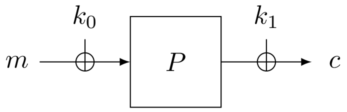


Figure 6.3: One-round key-alternating cipher: The Even-Mansour construction

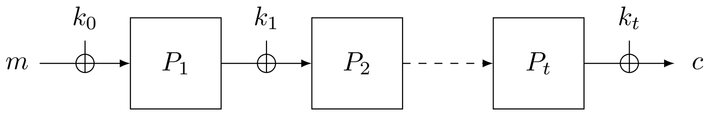


Figure 6.4: Key-alternating cipher with t rounds

construction is depicted in Figure 6.3. Even and Mansour proved that in order to have a reasonable success probability in decrypting a previously unqueried message, an attacker has to make at least about $2^{n/2}$ queries to the cipher, the permutation P, and their inverses. Daemen [52] showed that this bound is tight by presenting a differential attack on the Even-Mansour scheme that allows to successfully recover the key with a good probability after $2^{n/2}$ evaluations of both the permutation P and the encryption oracle.

6.5.2 The construction

In this section, we define the model of a key-alternating cipher (for a definition, see Section 2.2.2) we consider for our analysis.

For the key-alternating cipher with t rounds, we model the round functions as public, randomly chosen n-bit permutations P_i, with $t+1$ independent round keys k_i. In other words, for t n-bit permutations $P_1, \ldots, P_t$, $(t \geq 1)$, and $t+1$ n-bit keys $k_0, \ldots, k_t$, the n-bit block cipher $E = E_{k_0,\ldots,k_t}$ is defined by

$$E(x) \overset{\text{def}}{=} E_{k_0 \cdots k_t}(x) = P_t(\ldots P_2(P_1(x \oplus k_0) \oplus k_1) \ldots) \oplus k_t. \tag{6.24}$$

(See [33].) The construction is illustrated in Figure 6.4.

6.6 Generic attacks

In this section, we detail some generic (i.e., black box) attacks that apply to the key-alternating cipher in general.

6.6.1 Daemen's attack for $t = 1$

For the original Even-Mansour construction (in our setting, this corresponds to $t = 1$), a differential attack has been published by Daemen [52] meeting the lower bound of $2^{n/2}$ evaluations of P proven by Even and Mansour. It can be described as follows:

1. Choose s plaintext pairs (m_i, m_i^*), $1 \leq i \leq s$, with $m_i \oplus m_i^* = \Delta$ for any nonzero constant Δ.
2. Get the encryptions (c_i, c_i^*) of the s pairs.
3. For $2^n/s$ values v:
    (a) Compute $w' := P(v) \oplus P(v \oplus \Delta)$.
    (b) If $w' = c_i \oplus c_i^*$ for some i: Output $k_0 := v \oplus m_1$ and $k_1 := c_1 \oplus P(m_1 \oplus k_0)$ and stop.

For a random permutation P, only very few values of v are expected to satisfy $P(v) + P(v + \Delta) = c_i \oplus c_i^*$. The wrong candidates can be easily filtered in step (3b) by testing them on a few additional encryptions. After encrypting s plaintext pairs, one has to perform about $2 \cdot 2^n/s$ evaluations of P. The expression $2(s+2^n/s)$ is minimal for $s = 2^{n/2}$. In this case, the time complexity is $2^{n/2}$ with a storage requirement of $2^{n/2}$ plaintext pairs.

6.6.2 A meet in the middle attack

The t-round key-alternating cipher can be attacked using a basic meet in the middle technique which finds the $(t + 1)$ n-bit keys in time and space $2^{tn/2}$ for $t > 1$. We briefly outline this attack for the case $t = 2$. In this case, we want to recover the key (k_0, k_1, k_2).

1. Fix a pair of plaintexts (m_1, m_2). Compute the values $P_1(m_1 \oplus k_0) \oplus P(m_2 \oplus k_0)$ for all possible 2^n values of k_0 and save them in a sorted table T.
2. Obtain the encryptions c_1 and c_2 of m_1 respectively m_2.
3. For all 2^n possible values of k_2 compute $P_2^{-1}(c_1 \oplus k_2) \oplus P_2^{-1}(c_2 \oplus k_2)$ and look for a match in T.
4. Each match gives candidate values for (k_0, k_1, k_2), which can be tested against additional encryptions.

For $t = 2$, both time and storage complexities are 2^n. For general t, time and storage $2^{tn/2}$ are required.

6.7 Statistical properties

A fundamental cryptographic property of a block cipher is its Fourier spectrum that completely defines the cipher via the Fourier transform and whose distribution is closely related to the resistance against linear cryptanalysis [44].

To support security claims, block cipher designs are usually accompanied by arguments why these Fourier coefficients cannot take values exploitable by an attacker (see for instance, [3, 32, 57]). In most cases, however, formal proofs of these properties are technically infeasible and designers limit themselves to demonstrating upper bounds on linear trail probabilities, while usually the contributions of multiple trails has to be added to obtain the actual Fourier coefficients. Such an approach, while practical, does not allow for an accurate estimation of the data complexity of statistical attacks, that typically depends on numerous trails [105, 133], see also the discussion of the linear hull effect in Section 5.4.

In contrast to this, in this section we study key alternating ciphers by directly inspecting its Fourier coefficients. This provides a more informative analysis than for standard block ciphers, as we study the distribution of the Fourier coefficients for the cipher over all keys, as opposed to bounding the mean value of this distribution. This is made possible by the use of fixed public permutations in our construction. More precisely, in a key-alternating cipher using $t \geq 2$ fixed public permutations, we study the distribution of the Fourier coefficients over all cipher keys. By comparing this distribution to that over all permutations, we can determine and even quantify to which extent the key-alternating construction is theoretically sound from the perspective of Fourier analysis.

In this context, it is especially interesting to determine the effect of the number of rounds in a key-alternating cipher. Folklore suggests that more rounds imply higher security. Whether and to which extent this is true is subject to the analysis described in this section.

6.7.1 Fourier coefficients and Fourier spectrum

The Fourier spectrum has been defined in Section 2.1. For a block cipher F, we denote the Fourier coefficient at point (α, β) as $W^F_{\alpha,\beta}[K]$ to emphasize its dependency on key K. If F is the t-round key-alternating cipher, this is denoted by $W_{\alpha,\beta}^{P_1,\dots,P_t}[K]$.

The Fourier coefficient at point (α, β) is related to the correlation of the linear approximation (α, β) via $C_{\alpha,\beta} = 2^{-n} W_{\alpha,\beta}$. The following characterisation for the distribution of Fourier coefficients in a Boolean permutation is a reformulation of Theorem 5.7 in terms of Fourier coefficients:

Theorem 6.17 ([60, Corollary 4.3, Lemma 4.6])**.** *When $n \geq 5$, the distribution of the Fourier coefficient W_{α_0,β_0}^{P} with $\alpha_0, \beta_0 \neq 0$ over all n-bit permutations can be approximated by the following distribution up to continuity correction:*

$$W_{\alpha_0,\beta_0}^{P} \sim_P \mathcal{N}(0, 2^n). \tag{6.25}$$

The distribution of Theorem 6.17 is the reference point throughout the section: A block cipher cannot have a better distribution of Fourier coefficients than that close to Theorem 6.17.

6.7.2 Fourier coefficients in the single-round key-alternating cipher

Let F be the key-alternating cipher with $t = 1$, corresponding to the single-round Even-Mansour cipher, that is, a fixed public permutation P bracketed by two key additions k_0 and k_1, respectively (see Figure 6.3). Writing W_{β_0,β_1}^{P} for the Fourier coefficient for the underlying permutation P at point (β_0, β_1), the Fourier coefficient for the whole cipher at this point is given by

$$W_{\beta_0,\beta_1}^{F} = (-1)^{\beta_0^T k_0 \oplus \beta_1^T k_1} W_{\beta_0,\beta_1}^{P}. \tag{6.26}$$

Now we determine the distribution of W_{β_0,β_1}^{F} ($\beta_0 \neq 0$, $\beta_1 \neq 0$) over all cipher keys (k_0, k_1).

From (6.26), it is immediate that only the factor $(-1)^{\beta_0^T k_0 \oplus \beta_1^T k_1}$, namely the sign of the expression, varies over the keys. Therefore, we obtain a distribution with a support containing only two elements: $-W_{\beta_0,\beta_1}^{P}$ and W_{β_0,β_1}^{P}. This means that the value of the cipher's Fourier coefficient W_{β_0,β_1}^{F} only varies in its sign from key to key.

Note that this is structurally very different from the distribution of the Fourier coefficients over all permutations from Theorem 6.17. We conclude that the statistical behaviour of the single-round Even-Mansour construction can easily be distinguished from a random permutation.

6.7.3 Fourier coefficients in the t-round key-alternating cipher

In this section, we study the distribution of the Fourier coefficients over the keys for the general key-alternating construction with $t \geq 2$ rounds. Before stating our main result on the statistical analysis of key-alternating ciphers, we need a technical lemma:

Lemma 6.18. *Consider two independent random variables X_1 and X_2 having standard normal distributions $\mathcal{N}_1(0,1)$ and $\mathcal{N}_2(0,1)$, respectively. Then mean and variance of their product $Z = X_1 \cdot X_2$ are given by $\mu_Z = 0$ and $\sigma_Z^2 = 1$.*

Proof. The probability density functions of X_1 and X_2 are $\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x_1^2}$ and $\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x_2^2}$, respectively. Since they are continuous, the moment generating function of the distribution of Z is given by the Riemann integral

$$M_Z(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-\frac{1}{2}x_1^2-\frac{1}{2}x_2^2}e^{tx_1x_2}\,dx_1\,dx_2 = \frac{1}{\sqrt{1-t^2}}.$$

Expanding the logarithm of $M_Z(t)$ in a power series in t, we find

$$\ln(M_Z(t)) = \sum_{n=0}^{\infty} m_n \frac{t^n}{n!} = \sum_{k=1}^{\infty}\frac{1}{2k}t^{2k} = \frac{1}{2}t^2 + \frac{1}{4}t^4 + \cdots,$$

and, therefore, $\mu_Z = m_1 = 0$ and $\sigma_Z^2 = m_2 = 1$, as claimed. □

Theorem 6.19. *Fix a point (β_0, β_t) with $\beta_0, \beta_t \neq 0$ in the Fourier spectrum of the t-round key-alternating n-bit block cipher with round permutations $P_1, \ldots, P_t$ for $t \geq 2$ and sufficiently high n. Then the distribution of the Fourier coefficient $W_{\beta_0,\beta_t}^{P_1,\ldots,P_t}$ at this point over all keys K is approximated by:*

$$W_{\beta_0,\beta_t}^{P_1,\ldots,P_t}[K] \sim_K \mathcal{N}\left(0, (1+\varepsilon)\left(\frac{2^n-1}{2^n}\right)^{t-1} 2^n\right), \tag{6.27}$$

assuming that the distributions over points of the Fourier spectra of the permutations P_i, $1 \leq i \leq t$, have variances satisfying

$$\underset{(\beta_{i-1},\beta_i)}{\mathbf{Var}}\left[W_{\beta_{i-1},\beta_i}^{P_i}\right] \geq 2^{n/2}, \tag{6.28}$$

and that for any given key K, the signs of the Fourier coefficients behave independently for different points. The deviation of the permutations P_i from the mean over all permutations Q_i is quantified by factor $(1+\varepsilon)$:

$$\begin{aligned} &\textstyle\sum_{(\beta_1,\ldots,\beta_{t-1})}\left(W_{\beta_0,\beta_1}^{P_1}\cdots W_{\beta_{t-1},\beta_t}^{P_t}\right)^2 \\ &= (1+\varepsilon)\cdot \mathbf{E}_{Q_1,\ldots,Q_t}\left[\textstyle\sum_{(\beta_1,\ldots,\beta_{t-1})}\left(W_{\beta_0,\beta_1}^{Q_1}\cdots W_{\beta_{t-1},\beta_t}^{Q_t}\right)^2\right]. \end{aligned} \tag{6.29}$$

Proof. Consider a fixed point (β_0, β_t), $\beta_0, \beta_t \neq 0$, in the Fourier spectrum for the t-round key-alternating cipher with keys $K := (k_0, \ldots, k_t)$. Denote by β_i, $1 \leq i < t$, the intermediate selection pattern at the addition of k_i, and set $\beta := (\beta_1, \ldots, \beta_{t-1})$ and $\Gamma := (\beta_0, \ldots, \beta_t)$. By the theorem of trail composition (Theorem 7.8.1 in [57]), we have

$$W^{P_1,\ldots,P_t}_{\beta_0,\beta_t}[K] = 2^{n(1-t)} \sum_{\beta} W^{P_1}_{\beta_0,\beta_1} \cdots W^{P_t}_{\beta_{t-1},\beta_t} \cdot (-1)^{\Gamma^T K}, \tag{6.30}$$

with $W^{P_i}_{\beta_{i-1},\beta_i}$ denoting the Fourier coefficient of P_i at point (β_{i-1}, β_i). For each $\beta \neq 0$, define the random variable X_β as

$$X_\beta \overset{\text{def}}{=} W^{P_1}_{\beta_0,\beta_1} \cdots W^{P_t}_{\beta_{t-1},\beta_t} \cdot (-1)^{\Gamma^T K}, \tag{6.31}$$

so that

$$W^{P_1,\ldots,P_t}_{\beta_0,\beta_t}[K] = \sum_{\beta} X_\beta. \tag{6.32}$$

If, for any given key K, the quantities $\Gamma^T K$ behave independently over different β, as assumed in the claim of the theorem, we have that

$$X_\beta \sim_K W^{P_1}_{\beta_0,\beta_1} \cdots W^{P_t}_{\beta_{t-1},\beta_t} \cdot (-1)^r, \tag{6.33}$$

with $r \sim \text{Bern}(\frac{1}{2})$, where the distribution is taken over the keys, and $\text{Bern}(p)$ denotes the Bernoulli distribution with success probability p.

Note that $\mathbf{E}[X_\beta] = \frac{1}{2}(W^{P_1}_{\beta_0,\beta_1} \cdots W^{P_t}_{\beta_{t-1},\beta_t} - W^{P_1}_{\beta_0,\beta_1} \cdots W^{P_t}_{\beta_{t-1},\beta_t}) = 0$. The variance of X_β is given by

$$\begin{aligned}\mathbf{Var}[X_\beta] &= \frac{1}{2}\left(W^{P_1}_{\beta_0,\beta_1} \cdots W^{P_t}_{\beta_{t-1},\beta_t}\right)^2 + \frac{1}{2}\left(-W^{P_1}_{\beta_0,\beta_1} \cdots W^{P_t}_{\beta_{t-1},\beta_t}\right)^2 \\ &= \left(W^{P_1}_{\beta_0,\beta_1} \cdots W^{P_t}_{\beta_{t-1},\beta_t}\right)^2.\end{aligned}$$

Furthermore, with $b := 2^{tn} + 1$, we have

$$\lim_{m \to \infty} \Pr(|X_m| < b) = 1, \tag{6.34}$$

as each of the t multiplicands $W^{P_i}_{\beta_{i-1},\beta_i}$ of X_m are bounded by 2^n. On the other hand, the variance of all partial sums is unbounded by assumption (6.28) that $\mathbf{Var}_{\beta_{i-1},\beta_i}\left[W^{P_i}_{\beta_{i-1},\beta_i}\right] \geq 2^{n/2}$ and a standard comparison test:

$$\lim_{m\to\infty} \sum_{i=1}^{m} 2^{n/2} = \infty \Longrightarrow \lim_{m\to\infty} \mathbf{Var}\left[\sum_{i=1}^{m} X_i\right] = \lim_{m\to\infty} \sum_{i=1}^{m} \mathbf{Var}\left[X_i\right] = \infty. \tag{6.35}$$

A sequence of independent (one can consider the X_β as independent since the signs are independent) random variables fulfilling (6.34) and (6.35) obeys the Chebyshev formulation of the central limit theorem (Theorem 2.1. (Note that though we operate with finite numbers of summands, the conditions at infinity have to be checked for any application of the central limit theorem.) Therefore, we have the following approximation, since the number of summands is high (it is exponential in n and in all interesting cases $n \geq 32$):

$$\sum_\beta X_\beta \sim_K \mathcal{N}(0, s^2) \tag{6.36}$$

with $s^2 := \sum_\beta \mathbf{Var}[X_\beta]$. The mean of s^2 over all permutations $Q_1, \ldots, Q_t$ can now be determined as $\mathbf{E}_{Q_1,\ldots,Q_t}[s^2] = \mathbf{E}_{Q_1,\ldots,Q_t}\left[\sum_\beta \left(W^{Q_1}_{\beta_0,\beta_1} \cdots W^{Q_t}_{\beta_{t-1},\beta_t}\right)^2\right] = \sum_\beta \mathbf{E}_{Q_1,\ldots,Q_t}\left[\left(W^{Q_1}_{\beta_0,\beta_1} \cdots W^{Q_t}_{\beta_{t-1},\beta_t}\right)^2\right] = \sum_\beta \mathbf{Var}_{Q_1,\ldots,Q_t}\left[W^{Q_1}_{\beta_0,\beta_1} \cdots W^{Q_t}_{\beta_{t-1},\beta_t}\right] + \left(\mathbf{E}_{Q_1,\ldots,Q_t}\left[W^{Q_1}_{\beta_0,\beta_1} \cdots W^{Q_t}_{\beta_{t-1},\beta_t}\right]\right)^2$ by linearity of expectation and definition of variance. By Theorem 6.17, $W^{Q_i}_{\beta_{i-1},\beta_i} \sim_{Q_i} \mathcal{N}(0, 2^n) = 2^{n/2}\mathcal{N}(0,1)$ for each i, so $W^{Q_1}_{\beta_0,\beta_1} \cdots W^{Q_t}_{\beta_{t-1},\beta_t} \sim 2^{t(n/2)}\mathcal{N}(0,1) \cdots \mathcal{N}(0,1)$, where the product is over t standard normal distributions. The mean of this distribution is zero, and by Lemma 6.18, the variance of the product of two independent standard normal distributions $Z := \mathcal{N}(0,1)\mathcal{N}(0,1)$ is given by $\mathbf{Var}[Z] = 1$. The same applies to $t > 2$. Consequently, $\mathbf{Var}_{Q_1,\ldots,Q_t}\left[W^{Q_1}_{\beta_0,\beta_1} \cdots W^{Q_t}_{\beta_{t-1},\beta_t}\right] = \left(2^{t(n/2)}\right)^2 \cdot 1 = 2^{tn}$ for each β. Note that we have $(2^n - 1)^{t-1}$ values of β with no $\beta_i = 0$, so $\mathbf{E}[s^2] = (2^n - 1)^{t-1} 2^{nt}$.

Recall from (6.29) that for the t-round cipher with the permutations $P_1, \ldots, P_t$, we have that $s^2 = \sum_\beta \mathbf{Var}[X_\beta] = (1 + \varepsilon)\mathbf{E}_{Q_1,\ldots,Q_t}[s^2]$. The distribution of $W^{P_1,\ldots,P_t}_{\beta_0,\beta_t}$ over all keys is therefore given by

$$\begin{aligned} W^{P_1,\ldots,P_t}_{\beta_0,\beta_t} &\sim_K 2^{n(1-t)}\mathcal{N}(0, (1+\varepsilon)(2^n - 1)^{t-1} 2^{nt}) \\ &= \mathcal{N}(0, (1+\varepsilon)(2^n - 1)^{t-1} 2^{2n-nt}) \\ &= \mathcal{N}(0, (1+\varepsilon)\left(\frac{2^n - 1}{2^n}\right)^{t-1} 2^n), \end{aligned} \tag{6.37}$$

as claimed. □

6.7.4 Discussion

Theorem 6.19 basically says that once the fixed underlying permutations of a t-round key-alternating cipher ($t \geq 2$) are close to average (which is the case for randomly drawn permutations with high probability), the distribution of Fourier coefficients for the key-alternating cipher over all keys becomes close to that over all permutations. This is in stark contrast to the single-round Even-Mansour cipher, where the distribution over the keys was structurally different.

We require condition (6.28) essentially to ensure that we sum over sufficiently many possible selection patterns for β such that we can invoke the central limit theorem. This in particular excludes the trivial case where all P_i are linear, in which their variances would be zero, and the sum in (6.30) would only have one summand.

The required notion of "close to average" is expressed in the factor $1 + \varepsilon$ in equation (6.29) and essentially means that the sum of the squared Fourier coefficients over all trails has to be within $(1 + \varepsilon)$ of its mean over all permutations.

Interestingly, the latter deviation ε from the mean in (6.29) is small for most choices of the P_i. For instance, in case $t = 2$, it can be shown that over all permutations, mean and variance of each summand in (6.29) are 2^{2n} and 2^{4n+2}, respectively. The whole sum then approximately follows a normal distribution $\mathcal{N}(2^{3n} - 2^{2n}, 2^{5n+2} - 2^{4n+2})$. This means that for *randomly drawn permutations* P_1, P_2, the sum $\sum_{\beta_1} \left(W^{P_1}_{\beta_0,\beta_1} W^{P_2}_{\beta_1,\beta_2}\right)^2$ will be within d standard deviations from its mean with probability $\operatorname{erf}\left(d/\sqrt{2}\right)$. Notably, this implies $\Pr(|\varepsilon| \leq 2^{-n/2+3}) \approx 0.9999$, i.e. $|\varepsilon|$ only very rarely exceeds $2^{-n/2+3}$.

While this theorem emphasizes the constructive effect of having two and more rounds in a key-alternating cipher, we observe that the exponent $t - 1$ in the variance of the resulting distribution of Fourier coefficients over the keys indicates that excessively increasing the number of rounds t will result in a smaller and smaller variance, which at some point will result in a noticable difference from the reference distribution of Theorem 6.17.

Theorem 6.19 gives the distribution over all keys of the Fourier coefficient $W^{P_1,\ldots,P_t}_{\beta_0,\beta_t}$ individually for each nontrivial point (β_0, β_t). Appropriate choices for the P_i should have distributions close to $\mathcal{N}(0, 2^n)$ for each nontrivial point, not only for some of them. Conversely, the distribution of the Fourier coefficient at the (trivial) point $(\beta_0, 0)$ differs from (6.27) for any choice of the P_i, since it is constant over the keys.

Note also that the result of Theorem 6.19 does not require the underlying permutations to be different. Moreover, it does not require the permutations P_i to be randomly drawn from the set of all permutations, but holds for any fixed choice of permutations satisfying (6.28). To obtain a distribution close to ideal, however, the set of underlying permutations has to ensure a small deviation ε in (6.29). As argued above, drawing the underlying permutations at random from the set of all permutations is highly likely to result in a very small deviation ε from the average.

Summarising, the results of Theorem 6.19 suggest that once the $t \geq 2$ underlying permutations are carefully chosen and fixed, the t-round key-alternating cipher for each secret key is likely to be statistically sound with regard to linear attacks. More precisely, the distributions of the Fourier coefficients for the t-round key-alternating cipher over all keys become close to those over all permutations.

6.8 Indistinguishability analysis of key-alternating ciphers

Complementing our analysis of the statistical behaviour of the key-alternating cipher, we describe a study of its indistinguishability from a (pseudo-)random permutation in this section. As outlined in Section 6.5, this is an important open question in symmetric-key cryptography. Our discussion is based on the publication [33].

6.8.1 The provable security setting

The provable security analysis of the t-round key-alternating construction is carried out in the following setting: Considering the t randomly chosen permutations P_i and the resulting t-round cipher defined by (6.24) as oracles that A can query as blackboxes, we are interested in how many queries to the oracles A will need to distinguish this construction from a random permutation. We consider information-theoretic adversaries that are limited to making a certain number of queries, but can use unlimited computational and storage resources. Note that this constitutes a very strong adversarial model.

For the remainder of this chapter, we declare $N = 2^n$. The provable security setting is then defined as follows.

Definition 6.20 ([33])**.** The PRP (pseudorandom permutation) security of the t-round key-alternating cipher E against an adversary querying the t

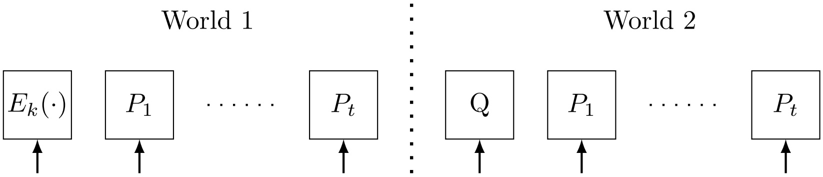


Figure 6.5: The indistinguishability model for the key-alternating cipher [33].

permutations P_t and the encryption oracle $E(\cdot)$ and their inverses is defined as

$$\mathbf{Adv}^{\text{PRP}}_{E,N,t}(A) = \Pr[k_0 \cdots k_t \leftarrow \{0,1\}^n; A^{E_{k_0 \cdots k_t}, P_1, \ldots, P_t} = 1] - \Pr[A^{Q, P_1, \ldots, P_t} = 1],$$

where $Q, P_1, \ldots, P_t$ are independent permutations sampled uniformly at random in each experiment.

The maximum advantage over all adversaries A limited to q queries is denoted

$$\mathbf{Adv}^{\text{PRP}}_{E,N,t}(q) = \max_A \mathbf{Adv}^{\text{PRP}}_{E}(A).$$

Graphically, the attacker has to distinguish the two worlds depicted in Figure 6.5

6.8.2 An indistinguishability bound for key-alternating ciphers

A bound for $\mathbf{Adv}^{\text{PRP}}_{E,N,t}(q)$ can be obtained by what is known as a *hybrid argument* [79]: Instead of directly evaluating the distinguishing advantage between two worlds (more formally, distributions) D_0 and D_2, a hybrid world D_1 is introduced which allows individually upper-bounding the adversary's advantage of distinguishing between D_0 and D_1, and D_1 and D_2. Adding up the two advantages then yields an upper bound for the original problem.

In the case of the PRP security of key-alternating ciphers, this can be applied as follows [33]: The hybrid is an oracle $\tilde{O}(N,t)$ using a sampling procedure for answering the queries to E and $P_1, \ldots, P_t$ that is slightly modified in comparison to world one. The following individual bounds are obtained:

Proposition 6.21. *Let $q < N/100$. With $\tilde{O}(N,t)$ the hybrid as above,*

$$\begin{aligned} &\Pr[k_0 \cdots k_t \leftarrow \{0,1\}^n; A^{E_{k_0 \cdots k_t}, P_1, \ldots, P_t} = 1] \\ &\qquad\qquad - \Pr[k_0, \ldots, k_t \leftarrow \{0,1\}^n; A^{\tilde{O}(N,t)} = 1] \leq \frac{4.3 q^3 t}{N^2} \end{aligned}$$

for every adversary A making at most q queries.

Proposition 6.22. *Let* $q = N^{\frac{t}{t+1}}/Z$ *for some* $Z \geq 1$ *be such that* $q < N/3$. *With* $\tilde{O}(N,t)$ *as above,*

$$\Pr[k_0, \ldots, k_t \leftarrow \{0,1\}^n; A^{\tilde{O}(N,t)} = 1] \\ - \Pr[A^{Q,P_1,\ldots,P_t} = 1] \leq \frac{t+1}{Z^{t+1}}.$$

for every adversary A *making at most* q *queries.*

Adding these inequalities yields the sought upper bound for $\mathbf{Adv}^{\mathrm{PRP}}_{E,N,t}(q)$:

Theorem 6.23. *Let* $N = 2^n$ *and let* $q = N^{\frac{t}{t+1}}/Z$ *for some* $Z \geq 1$. *Then, for any* $t \geq 1$*, and assuming* $q < N/100$*, we have*

$$\boldsymbol{Adv}^{\mathrm{PRP}}_{E,N,t}(q) \leq \frac{4.3q^3t}{N^2} + \frac{t+1}{Z^t}.$$

Observe that for $t \geq 2$, the limiting term in the above bound is $4q^3t/N^2$, and $4q^3t/N^2 = c$ for any constant $0 < c < \frac{1}{2}$ implies $q \approx N^{2/3}$. Therefore, a lower bound on the number of queries that an information-theoretic adversary needs to make in order to have non-negligible advantage in the PRP security game is $q \approx N^{2/3}$.

This has to be compared against $q \approx N^{1/2}$ in the case of $t = 1$, the original Even-Mansour construction. We however conjecture that this bound is not optimal for $t > 2$, and that the real lower bound is increasing with t:

Conjecture 6.24. *Let* $t \geq 1$. *For a* t*-round key-alternating cipher with block length* n*, an attacker needs to make at least about*

$$q \approx 2^{\frac{t}{t+1}n}$$

queries before being able to distinguish the encryption oracle from a random permutation with non-negligible probability.

The result of Even-Mansour [70] and Theorem 6.23 imply that this conjectured bound is tight for $t = 1, 2$.

6.8.3 An attack meeting the conjectured bound

In this section, we demonstrate that there is a generic attack on the t-round key-alternating cipher with a query complexity meeting the proved bound for $t = 2$ and the bound of of Conjecture 6.24 for any $t \geq 2$. This implies that the intrinsic query complexity of distinguishing this cipher from a random permutation is upper bounded by the conjectured bound $2^{\frac{t}{t+1}n}$.

Similar to the proof of Proposition 6.22, the attack verifies key candidates by attempting to construct a sequence of connected queries to the permutations P_i that contradicts the corresponding query to the encryption oracle. It can be described as follows [33]:

1. Make q queries to E_k and store them in the list $\mathcal{M}$. Also make q queries to each of the permutations P_1 to P_t and store the queries to P_i in a list $\mathcal{P}_i$.
2. For each key candidate $(k_0, k_1, \ldots, k_t)$ do:
    (a) For each $x_1 \in \mathcal{M}$, find all sequences of values $(x_1, \ldots, x_{t-1})$ such that and $x_i \oplus k_{i-1} \in \mathcal{P}_i$, $\forall 1 \leq i \leq t$ and $P_i(x_i \oplus k_{i-1}) = x_{i+1}$, $\forall 1 \leq i \leq t-1$.

    (b) Check if

    $$P_t(x_t \oplus k_{t-1}) \oplus k_t = E(x_1) \tag{6.38}$$

    for all these sequences.

    (c) If (6.38) holds for all sequences, suggest $(k_0, k_1, \ldots, k_t)$ as correct value of the key;

    (d) otherwise, it is certainly the wrong value of the key.

Note that for any value of $q \geq 1$, the right key will be suggested, whereas increasing q will have the effect of reducing the number of false positives.

In order to determine how many queries are needed to determine the key with good probability, we note that if the queries to E_k are random and non-repeating, the total number s of sequences satisfying all conditions $x_k \oplus k_{i-1} \in \mathcal{P}_i$ in step (a) will be about $s = q \cdot \left(\frac{q}{2^n}\right)^t = \frac{q^{t+1}}{2^{tn}}$. Over all key candidates, each will fulfill (6.38) with probability about 2^{-n}, therefore the probability of a key being suggested is roughly $(2^{-n})^s$. To obtain a good reduction of the $2^{(t+1)n}$ key candidates, a small multiple of $q = 2^{\frac{t}{t+1}n}$ queries are sufficient. In total, this implies a complexity of about $t \cdot 2^{\frac{t}{t+1}n}$ queries to all oracles, which meets the conjectured bound.

Also note that this not an attack in the practical meaning of the word, since the time (as opposed to the query complexity) exceeds $2^{(t+1)n}$.

6.9 Practically instantiating the construction

In this section, we discuss how the t-round key-alternating cipher can be practically instantiated with t public permutations.

A natural approach to building a practical cipher following the t-permutation construction is to base the t fixed permutations on a block cipher by fixing some keys. With $t = 1$, this corresponds to the original Even-Mansour construction, so the security level is limited to $2^{n/2}$ operations with n denoting the cipher's block length. We therefore require $t > 1$.

In the following we describe a sample construction with $t = 2$, that is, we consider the 2-round key alternating construction with permutations P_1 and P_2 and the keys k_0, k_1, k_2.

6.9.1 AES^2: a block cipher proposal based on AES

The construction is defined by fixing two randomly chosen 128-bit AES-128 keys, which specifies the permutations P_1 and P_2. The key is comprised by three independently chosen 128-bit secret keys k_0, k_1, k_2.

Let AES$[k]$ denote the (10-round) AES-128 algorithm with the 128-bit key k and the 128-bit quantities π_1, π_2 be defined based on the first 256 bits of the binary digit expansion of $\pi = 3.1415\ldots$:

$$\pi_1 := \texttt{0x243f6a8885a308d313198a2e03707344} \quad \text{and}$$

$$\pi_2 := \texttt{0xa4093822299f31d0082efa98ec4e6c89}.$$

Then we denote the resulting 2-permutation construction by AES$^2[k_0, k_1, k_2]$. Its action on the 128-bit plaintext m is defined as:

$$\mathrm{AES}^2[k_0, k_1, k_2](m) := \mathrm{AES}[\pi_2](\mathrm{AES}[\pi_1](m \oplus k_0) \oplus k_1) \oplus k_2. \tag{6.39}$$

On the security of AES^2

The indistinguishability result of Theorem 6.23 implies that any attack on AES^2 in the single secret-key model with complexity below $2^{128 \cdot 2/3} \approx 2^{85}$ will necessarily have to exploit properties of AES with the fixed known keys in a non-black box way.

	Intel Xeon X5670 2.93 GHz, 12 MB L3 cache	Intel Core i7 640M 2.8 GHz, 4 MB L3 cache
AES^2, AES-NI, ECB	2.54 cpb	2.69 cpb
AES^2, AES-NI, CTR	2.65 cpb	2.76 cpb
AES-128, AES-NI, ECB	1.18 cpb	1.25 cpb
AES-128, AES-NI, CTR	1.32 cpb	1.36 cpb
AES-128, bitsliced, CTR	7.08 cpb	7.84 cpb
AES-128, OpenSSL, CTR	15.73 cpb	16.76 cpb

Table 6.3: Practical performance of AES^2 using AES-NI instructions.

At the same time, to the best of our knowledge, the best known way to attack AES^2 would be the meet-in-the middle attack of Section 6.6.2 with a complexity of 2^{128} time and 2^{128} data.

Concerning statistical attacks, if the distribution of Fourier coefficients for $\text{AES}[\pi_1]$ and $\text{AES}[\pi_2]$ meets the assumption of average behaviour required by Theorem 6.19, this theorem suggests that the Fourier coefficients for AES^2 are distributed close to ideal, implies resistance against linear cryptanalysis and some of its variants.

Performance

AES^2 can be implemented very efficiently in software on general-purpose processors. Note that the two AES keys π_1 and π_2 are fixed and, therefore, the round keys for the two AES transformations can be precomputed.

On the Westmere architecture generation of Intel general-purpose processors, AES^2 can be implemented using the AES-NI instruction set [80]. As the AES round instructions are pipelined, we fully utilise the pipeline by processing four independent plaintext blocks in parallel implementing the basic electronic codebook mode (ECB) and counter mode (CTR). The performance of these implementations on recent processors is demonstrated and compared to two conventional implementations of AES-128 (i.e. without AES-NI instructions) – the bitsliced implementation of [99] and the OpenSSL 1.0.0e implementation based on lookup tables. The results are summarised in Table 6.3. All numbers are given in cycles per byte (cpb).

It turns out that on both platforms, the performance of AES^2 is almost equal to half that of AES, indicating that the overhead is very low. Compared to the best implementations of the AES which are in widespread use now on standard platforms, AES^2 provides a performance improvement of almost factor three and higher with the AES-NI instruction set.

6.10 Conclusions

In this chapter, we have studied two problems related to the design of block ciphers.

First, we demonstrated in Section 6.2 how to obtain a statement over the distribution of correlation of linear approximations and cardinality of differentials for one randomly drawn Boolean function or permutation, taking the probabilities over all linear approximations or differentials and showed that this has important applications in symmetric-key cryptography.

Second, we studied key-alternating ciphers from a theoretical point of view. By modeling the round functions as public random permutations, a lower bound of $2^{2n/3}$ on the number of queries an attacker needs to obtain in order to distinguish the construction from a random permutation was established in Section 6.8.

In Section 6.7, we gave an analysis of the security of this construction with respect to statistical attacks based on the distribution of its Fourier coefficients. This analysis revealed that under some plausible assumptions, the key-alternating construction can be considered sound with regard to linear attacks.

Chapter 7

Conclusion and outlook

> Jede Wissenschaft bedarf der
> Mathematik, die Mathematik
> bedarf keiner.
>
> Jakob Bernoulli, 1655–1705

In this chapter, we give a summary of the main results developed in this thesis and outline open problems.

7.1 Summary of results

This thesis aims at viewing the analysis and the design of symmetric-key algorithms: stream ciphers, hash functions, and block ciphers, from a mathematical perspective.

New cryptanalytic ideas modify our conception of how to design cryptosystems, and new designs inspire new techniques for attacking them. In Chapters 3 and 4, we looked at two different strategies to attack symmetric primitives, both based on ideas of how to solve some non-linear equations over finite fields of characteristic two.

First, in Chapter 3, we looked at this problem from the point of numerical optimisation. Translating the equations over $\mathbb{F}_2$ into a continuous optimisation problem over the reals, we proposed *nonsmooth cryptanalysis*, a technique aiming at avoiding real-valued expressions of higher degree by using nondifferentiable but Lipschitz-continuous functions. Applied to the stream cipher MICKEY from the eSTREAM final portfolio, we showed that this approach can successfully recover the key requiring only very little known plaintext, albeit with a total time complexity slightly inferior to brute force.

Chapter 4 was dedicated to the exploration of a discrete equation-solving strategy, the Rebound attack on hash functions [122]. Based on the analysis of the Wildpool design study, we presented and analysed the Whirlwind hash function which aims at taking the Rebound attack into account in its design. At the example of the Grøstl-0 hash function, we demonstrated that by extending the Rebound attack idea, the security margin of a hash function specifically designed to resist it, can be considerably lower than anticipated.

Even though any cryptanalysis with a complexity lower than brute force is considered a perfectly valid attack, the best attacks on real-world block ciphers very often require time and data complexities that make an experimental verification of the attack strategy infeasible. It is therefore extremely important to be able to accurately estimate these complexities using a mathematical model. In Chapter 5, we studied how to model and estimate the data and time complexities of linear cryptanalysis and differential attacks with structures more accurately by applying the theory of the probability distributions of correlations of linear approximations and cardinality of differentials over Boolean permutations [60]. This resulted in more accurate and realistic estimations for the complexities of these attacks. In the case of linear cryptanalysis, our analysis revealed that the long-standing assumption that using more data always improves the attack, does not hold in general.

Lastly, Chapter 6 studied two open problems related to the design of block ciphers. First, we demonstrated how to obtain a statement over the distribution of correlation of linear approximations and cardinality of differentials for one randomly drawn Boolean function or permutation, taking the probabilities over all linear approximations or differentials, providing us with a *reference point* for resistance against these attacks and some extensions such as zero-correlation and impossible differential cryptanalysis. Second, we studied key-alternating ciphers from a theoretical point of view. From both the perspectives of statistical attacks and indistinguishability from a random permutation, the constructive effect of having two or more rounds in a key-alternating cipher was proven.

7.2 Open problems

In the following, we list a couple of open research problems regarding the results and observations in this thesis.

In the context of nonsmooth cryptanalysis, a model for the estimation of the attack complexity when exceeding the capabilities of practical experimentation, is still lacking. Such an estimation would enable us to analyse the applicability of this equation solving technique to real-world ciphers also in the cases where only very few of the key bits have to be guessed.

Having explored rebound attacks, and some extensions of them, one can be fairly confident that both Whirlwind and Grøstl in their current versions resist these attacks with their proposed parameters. However, being able to prove something about resistance against rebound attacks along the same lines as we can prove resistance against basic differential attacks [56] would be a major step forward.

While the impact of the wrong-key randomisation hypothesis was studied in detail for linear cryptanalysis in this thesis, a similar study for differential cryptanalysis seems anything but straightforward. First of all, for the differential case, one has to avoid normal approximations to the Binomial distribution. Second, the resulting sum of Poisson distributions cannot be handled by an application of the Central Limit Theorem, since the sum of variances is not unbounded. A deeper mathematical analysis of this situation could provide an increase in our understanding of the complexity of differential attacks.

This situation is actually similar for the statistical analysis of key-alternating ciphers performed in Chapter 6. Unlike in linear cryptanalysis, for differential trails, there is no closed formula describing the relation between the expanded key bits and the individual probability contributions. Even though the existence of such a closed description might be considered doubtful, a similar statement to the one of Theorem 6.19 might be possible regardless of this characterisation.

Bibliography

[1] Masayuki Abe, editor. *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*. Springer, 2010.

[2] Martin R. Albrecht and Carlos Cid. Algebraic techniques in differential cryptanalysis. In Dunkelman [67], pages 193–208.

[3] R. Anderson, E. Biham, and L.R. Knudsen. A proposal for the Advanced Encryption Standard. NIST AES proposal, 1998.

[4] Steve Babbage and Matthew Dodd. The MICKEY stream ciphers. In Robshaw and Billet [146], pages 191–209.

[5] Adil M. Bagirov. A method for minimization of quasidifferentiable functions. *Optimization Methods and Software*, 17(1):31–60, 2002.

[6] Adil M. Bagirov, Gleb Beliakov, Musa A. Mammadov, Alexander M. Rubinov, and Julien Ugon. The GANSO programming library for global and non-smooth optimization. Centre for Informatics and Applied Optimization (CIAO), University of Ballarat. `http://guerin.ballarat.edu.au/ard/itms/CIAO/ganso/`.

[7] Adil M. Bagirov and A. A. Gasanov. A method of approximating a quasidifferential. *Russian Journal of Computational Mathematics and Mathematical Physics*, 35(4):403–409, 1995.

[8] Adil M. Bagirov and Julien Ugon. Piecewise partially separable functions and a derivative-free algorithm for large scale nonsmooth optimization. *Journal of Global Optimization*, 35:163–195, 2006.

[9] Thomas Baignères, Pascal Junod, and Serge Vaudenay. How far can we go beyond linear cryptanalysis? In Pil Joong Lee, editor, *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 432–450. Springer, 2004.

[10] Thomas Baignères and Serge Vaudenay. The complexity of distinguishing distributions (invited talk). In Reihaneh Safavi-Naini, editor, *ICITS*, volume 5155 of *Lecture Notes in Computer Science*, pages 210–222. Springer, 2008.

[11] Paulo S. L. M. Barreto, Ventzislav Nikov, Svetla Nikova, Vincent Rijmen, and Elmar Tischhauser. Whirlwind: a new cryptographic hash function. *Des. Codes Cryptography*, 56(2-3):141–162, 2010.

[12] Paulo S.L.M. Barreto and Vincent Rijmen. The Whirlpool hashing function. In *First open NESSIE Workshop, Leuven, Belgium*, 2000.

[13] Lynn M. Batten and Gleb Beliakov. Fast algorithm for the cutting angle method of global optimization. *Journal of Global Optimization*, 24:149–161, 2002.

[14] Gleb Beliakov and Julien Ugon. Implementation of novel methods of global and non-smooth optimization: GANSO programming library. *Optimization*, 56:543–546, 2007.

[15] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Keccak sponge function family main document. Submission to NIST SHA-3 competition (Round 3), 2011. `http://keccak.noekeon.org/`.

[16] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On the indifferentiability of the Sponge construction. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008.

[17] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The Keccak reference, version 3.0. Submission to NIST (SHA-3 competition, final round), 2011.

[18] D. N. Bessis and Frank H. Clarke. Partial subdifferentials, derivates and Rademacher's theorem. *Transactions of the American Mathematical Society*, 351(7):2899–2926, 1999.

[19] Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In Stern [155], pages 12–23.

[20] Eli Biham, Orr Dunkelman, and Nathan Keller. The rectangle attack - rectangling the Serpent. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 340–357. Springer, 2001.

[21] Eli Biham, Vladimir Furman, Michal Misztal, and Vincent Rijmen. Differential cryptanalysis of Q. In Mitsuru Matsui, editor, *FSE*, volume 2355 of *Lecture Notes in Computer Science*, pages 174–186. Springer, 2001.

[22] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *J. Cryptology*, 4(1):3–72, 1991.

[23] Eli Biham and Adi Shamir. Differential cryptoanalysis of Feal and N-Hash. In Davies [62], pages 1–16.

[24] Alex Biryukov, Christophe De Cannière, and Michaël Quisquater. On multiple linear approximations. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2004.

[25] Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors. *Selected Areas in Cryptography - 17th International Workshop, SAC 2010, Waterloo, Ontario, Canada, August 12-13, 2010, Revised Selected Papers*, volume 6544 of *Lecture Notes in Computer Science*. Springer, 2011.

[26] Alex Biryukov and Eyal Kushilevitz. Improved cryptanalysis of RC5. In Kaisa Nyberg, editor, *EUROCRYPT*, volume 1403 of *Lecture Notes in Computer Science*, pages 85–99. Springer, 1998.

[27] C. Blondeau and B.Gérard. Multiple differential cryptanalysis: Theory and practice (corrected). Technical Report 115, Cryptology ePrint Archive Report 2011/115, 2011.

[28] C. Blondeau and B. Gérard. The 561 differentials. private communication, 2011.

[29] Céline Blondeau and Benoît Gérard. Multiple differential cryptanalysis: Theory and practice. In Antoine Joux, editor, *FSE*, volume 6733 of *Lecture Notes in Computer Science*, pages 35–54. Springer, 2011.

[30] Céline Blondeau, Benoît Gérard, and Jean-Pierre Tillich. Accurate estimates of the data complexity and success probability for various cryptanalyses. *Des. Codes Cryptography*, 59(1-3):3–34, 2011.

[31] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full AES. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 344–371. Springer, 2011.

[32] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[33] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, François-Xavier Standaert, John Steinberger, and Elmar Tischhauser. Key-Alternating Ciphers in a Provable Setting: Encryption Using A Small Number of Public Permutations. In Thomas Johansson and David Pointcheval, editors, *Advances in Cryptology - EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 45–62, Cambridge,UK, 2012. Springer-Verlag.

[34] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, François-Xavier Standaert, John P. Steinberger, and Elmar Tischhauser. Key-alternating ciphers in a provable setting: Encryption using a small number of public permutations. *IACR Cryptology ePrint Archive*, 2012:35, 2012.

[35] Andrey Bogdanov and Vincent Rijmen. Linear hulls with correlation zero and linear cryptanalysis of block ciphers. *personal communication*, page 15, 2011.

[36] Andrey Bogdanov and Elmar Tischhauser. On the Wrong Key Randomization Hypothesis in Matsui's Algorithm 2, 2012.

[37] Julia Borghoff. *Cryptanalysis of Lightweight Ciphers.* PhD thesis, Technical University of Denmark, Kgs. Lynby, Denmark, 2010.

[38] Julia Borghoff, Lars R. Knudsen, and Krystian Matusiewicz. Hill climbing algorithms and Trivium. In Biryukov et al. [25], pages 57–73.

[39] Julia Borghoff, Lars R. Knudsen, and Mathias Stolpe. Bivium as a Mixed-Integer Linear Programming problem. In Matthew G. Parker, editor, *IMA Int. Conf.*, volume 5921 of *Lecture Notes in Computer Science*, pages 133–152. Springer, 2009.

[40] Endre Boros and Peter L. Hammer. Pseudo-Boolean optimization. *Discrete Applied Mathematics*, 123(1-3):155–225, 2002.

[41] R. K. Brayton, A. L. Sangiovanni-Vincentelli, C. T. McMullen, and G. D. Hachtel. *Logic Minimization Algorithms for VLSI Synthesis.* Kluwer Academic Publishers, 1984.

[42] Christophe De Cannière. Trivium: A stream cipher construction inspired by block cipher design principles. In Sokratis K. Katsikas, Javier Lopez, Michael Backes, Stefanos Gritzalis, and Bart Preneel, editors, *ISC*, volume 4176 of *Lecture Notes in Computer Science*, pages 171–186. Springer, 2006.

[43] Christophe De Cannière and Bart Preneel. Trivium. In Robshaw and Billet [146], pages 244–266.

[44] Florent Chabaud and Serge Vaudenay. Links between differential and linear cryptanalysis. In De Santis [63], pages 356–365.

[45] Joo Yeon Cho. Linear cryptanalysis of reduced-round PRESENT. In Pieprzyk [142], pages 302–317.

[46] Carlos Cid, Sean Murphy, and Matthew Robshaw. *Algebraic Aspects of the Advanced Encryption Standard.* Springer-Verlag, New York, 2006.

[47] Frank H. Clarke. *Optimization and Nonsmooth Analysis.* Wiley & Sons, New York, 1983.

[48] Baudoin Collard and François-Xavier Standaert. A statistical saturation attack against the block cipher PRESENT. In Marc Fischlin, editor, *CT-RSA*, volume 5473 of *Lecture Notes in Computer Science*, pages 195–210. Springer, 2009.

[49] Scott Contini, Arjen K. Lenstra, and Ron Steinfeld. VSH, an efficient and provable collision-resistant hash function. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 165–182. Springer, 2006.

[50] Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In Yuliang Zheng, editor, *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer, 2002.

[51] Yves Crama, Pierre Hansen, and Brigitte Jaumard. The basic algorithm for pseudo-Boolean programming revisited. *Discrete Applied Mathematics*, 29(2-3):171–185, 1990.

[52] Joan Daemen. Limitations of the Even-Mansour construction. In Imai et al. [92], pages 495–498.

[53] Joan Daemen. *Cipher and hash function design strategies based on linear and differential cryptanalysis.* PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium, 1995.

[54] Joan Daemen, René Govaerts, and Joos Vandewalle. Correlation matrices. In Preneel [143], pages 275–285.

[55] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. Linear frameworks for block ciphers. *Des. Codes Cryptography*, 22(1):65–87, 2001.

[56] Joan Daemen and Vincent Rijmen. The wide trail design strategy. In Bahram Honary, editor, *IMA Int. Conf.*, volume 2260 of *Lecture Notes in Computer Science*, pages 222–238. Springer, 2001.

[57] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES – The Advanced Encryption Standard.* Springer-Verlag, 2002.

[58] Joan Daemen and Vincent Rijmen. Probability distributions of correlation and differentials in block ciphers. Technical Report 212, IACR eprint Report 2005/212, 2005. `http://eprint.iacr.org/2005/212`.

[59] Joan Daemen and Vincent Rijmen. Plateau Characteristics and AES. *IET Information Security*, 1(1):11–17, 2007.

[60] Joan Daemen and Vincent Rijmen. Probability distributions of correlations and differentials in block ciphers. *Journal of Mathematical Cryptology*, 1(3):221–242, 2007.

[61] Joan Daemen and Vincent Rijmen. New criteria for linear maps in AES-like ciphers. *Cryptography and Communications*, 1(1):47–69, 2009.

[62] Donald W. Davies, editor. *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, volume 547 of *Lecture Notes in Computer Science.* Springer, 1991.

[63] Alfredo De Santis, editor. *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, volume 950 of *Lecture Notes in Computer Science.* Springer, 1995.

[64] Vladimir F. Demyanov. The rise of nonsmooth analysis: its main tools. *Kibernetika i Sistemnyi Analiz*, 4:63–85, Juliy-August 2002. Translated from Russian.

[65] Vladimir F. Demyanov and A. M. Rubinov. *Constructive Nonsmooth Analysis.* Verlag Peter Lang, Frankfurt am Main, 1995.

[66] Yvo Desmedt, editor. *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, volume 839 of *Lecture Notes in Computer Science*. Springer, 1994.

[67] Orr Dunkelman, editor. *Fast Software Encryption, 16th International Workshop, FSE 2009, Leuven, Belgium, February 22-25, 2009, Revised Selected Papers*, volume 5665 of *Lecture Notes in Computer Science*. Springer, 2009.

[68] Orr Dunkelman, Sebastiaan Indesteege, and Nathan Keller. A differential-linear attack on 12-round Serpent. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *INDOCRYPT*, volume 5365 of *Lecture Notes in Computer Science*, pages 308–321. Springer, 2008.

[69] Shimon Even and Yishay Mansour. A construction of a cipher from a single pseudorandom permutation. In Imai et al. [92], pages 210–224.

[70] Shimon Even and Yishay Mansour. A construction of a cipher from a single pseudorandom permutation. *J. Cryptology*, 10(3):151–162, 1997.

[71] H. Feistel. Cryptography and computer privacy. *Scientific American*, 228:15–23, 1973.

[72] FIPS PUB 46. DATA ENCRYPTION STANDARD (DES), 1976.

[73] Manlio Gaudioso, Enrico Gorgone, and M. F. Monaco. Piecewise linear approximations in nonconvex nonsmooth optimization. *Numerische Mathematik*, 113(1):73–88, 2009.

[74] P. Gauravaram, L.R. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schläffer, and S.S. Thomsen. Grøstl – a SHA-3 candidate. Submission to NIST (SHA-3 competition, Second round), 2008.

[75] P. Gauravaram, L.R. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schläffer, and S.S. Thomsen. Grøstl – a SHA-3 candidate. Submission to NIST (SHA-3 competition, final round), 2011.

[76] Carl Geiger and Christian Kanzow. *Numerische Verfahren zur Lösung unrestringierter Optimierungsaufgaben*. Springer-Verlag, Berlin, 1999.

[77] Henri Gilbert and Thomas Peyrin. Super-sbox cryptanalysis: Improved attacks for AES-like permutations. In Seokhie Hong and Tetsu Iwata, editors, *FSE*, volume 6147 of *Lecture Notes in Computer Science*, pages 365–383. Springer, 2010.

[78] Fred Glover, Bahram Alidaee, Cesar Rego, and Gary Kochenberger. One-pass heuristics for large-scale unconstrained binary quadratic problems. *European Journal of Operational Research*, 137:272–287, 2002.

[79] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[80] Shay Gueron. Intel Advanced Encryption Standard (AES) instructions set. Intel Mobility Group, Israel Development Center, Israel, 2010. `http://software.intel.com/file/24917`.

[81] Peter L. Hammer and Sergiu Rudeanu. *Boolean Methods in Operations Research and Related Areas.* Springer-Verlag, Berlin, 1968.

[82] Carlo Harpes, Gerhard G. Kramer, and James L. Massey. A generalization of linear cryptanalysis and the applicability of Matsui's piling-up lemma. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *EUROCRYPT*, volume 921 of *Lecture Notes in Computer Science*, pages 24–38. Springer, 1995.

[83] Philip Hawkes and Luke O'Connor. XOR and non-XOR differential probabilities. In Stern [155], pages 272–285.

[84] Tor Helleseth, editor. *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *Lecture Notes in Computer Science.* Springer, 1994.

[85] Miia Hermelin, Joo Yeon Cho, and Kaisa Nyberg. Multidimensional extension of Matsui's Algorithm 2. In Dunkelman [67], pages 209–227.

[86] Miia Hermelin and Kaisa Nyberg. Linear cryptanalysis using multiple linear approximations. In Pascal Junod and Anne Canteaut, editors, *Advanced Linear Cryptanalysis of Block and Stream Ciphers.* IOS Press, 2011.

[87] Jean-Baptiste Hiriart-Urruty and Claude Lemarechal. *Convex Analysis and Minimization Algorithms*, volume I and II. Springer-Verlag, Berlin, 1993.

[88] Jin Hong and Woo-Hwan Kim. TMD-Tradeoff and state entropy loss considerations of streamcipher MICKEY. In Subhamoy Maitra, C. E. Veni Madhavan, and Ramarathnam Venkatesan, editors, *INDOCRYPT*, volume 3797 of *Lecture Notes in Computer Science*, pages 169–182. Springer, 2005.

[89] K. Ideguchi, T. Owada, and H. Yoshida. A study on RAM requirements of various SHA-3 candidates on low-cost 8-bit CPUs, May 2009. `http://www.sdl.hitachi.co.jp/crypto/lesamnta/A_Study_on_RAM_Requirements.pdf`.

[90] Kota Ideguchi, Elmar Tischhauser, and Bart Preneel. Improved collision attacks on the reduced-round Grøstl hash function. In Mike Burmester, Gene Tsudik, Spyros S. Magliveras, and Ivana Ilic, editors, *ISC*, volume 6531 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2010.

[91] Kota Ideguchi, Elmar Tischhauser, and Bart Preneel. Internal Differential Collision Attacks on the Reduced-Round Grøstl-0 Hash Function. *Designs, Codes and Cryptography*, page 23, 2011. to appear.

[92] Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto, editors. *Advances in Cryptology - ASIACRYPT '91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan, November 11-14, 1991, Proceedings*, volume 739 of *Lecture Notes in Computer Science*. Springer, 1993.

[93] Sebastiaan Indesteege, Florian Mendel, Bart Preneel, and Christian Rechberger. Collisions and other non-random properties for step-reduced SHA-256. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography*, volume 5381 of *Lecture Notes in Computer Science*, pages 276–293. Springer, 2008.

[94] Jérémy Jean, María Naya-Plasencia, and Martin Schläffer. Improved analysis of ECHO-256. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography*, volume 7118 of *Lecture Notes in Computer Science*, pages 19–36. Springer, 2011.

[95] Michael J. Jacobson Jr., Vincent Rijmen, and Reihaneh Safavi-Naini, editors. *Selected Areas in Cryptography, 16th Annual International Workshop, SAC 2009, Calgary, Alberta, Canada, August 13-14, 2009, Revised Selected Papers*, volume 5867 of *Lecture Notes in Computer Science*. Springer, 2009.

[96] Pascal Junod. On the complexity of Matsui's attack. In Serge Vaudenay and Amr M. Youssef, editors, *Selected Areas in Cryptography*, volume 2259 of *Lecture Notes in Computer Science*, pages 199–211. Springer, 2001.

[97] Pascal Junod. On the optimality of linear, differential, and sequential distinguishers. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 17–32. Springer, 2003.

[98] Pascal Junod and Serge Vaudenay. Optimal key ranking procedures in a statistical cryptanalysis. In Thomas Johansson, editor, *FSE*, volume 2887 of *Lecture Notes in Computer Science*, pages 235–246. Springer, 2003.

[99] Emilia Käsper and Peter Schwabe. Faster and timing-attack resistant AES-GCM. In Christophe Clavier and Kris Gaj, editors, *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2009.

[100] Dmitry Khovratovich, Ivica Nikolic, and Christian Rechberger. Rotational rebound attacks on reduced Skein. In Abe [1], pages 1–19.

[101] Lars R. Knudsen. Practically secure Feistel cyphers. In Ross J. Anderson, editor, *FSE*, volume 809 of *Lecture Notes in Computer Science*, pages 211–221. Springer, 1993.

[102] Lars R. Knudsen. Truncated and higher order differentials. In Preneel [143], pages 196–211.

[103] Lars R. Knudsen. DEAL - a 128-bit block cipher. Technical Report 151, Department of Informatics, University of Bergen, 1998.

[104] Donald E Knuth. *The Art of Computer Programming - Seminumerical Algorithms*, volume 2. Addison-Wesley, 3rd edition edition, 1997.

[105] Xuejia Lai and James L. Massey. Markov ciphers and differential cryptanalysis. In Davies [62], pages 17–38.

[106] Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer. Rebound distinguishers: Results on the full whirlpool compression function. In Matsui [118], pages 126–143.

[107] Susan K. Langford and Martin E. Hellman. Differential-linear cryptanalysis. In Desmedt [66], pages 17–25.

[108] Gregor Leander. Small scale variants of the block cipher PRESENT. Technical Report 143, IACR eprint Report 2010/143, 2010. `http://eprint.iacr.org/2010/143`.

[109] Gregor Leander. On linear hulls, statistical saturation attacks, PRESENT and a cryptanalysis of PUFFIN. In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 303–322. Springer, 2011.

[110] F.C. Leone, R.B. Nottingham, and L.S. Nelson. The folded normal distribution. *Technometrics*, 3(4):543–550, 1961.

[111] Rudolf Lidl and Harald Niederreiter. *Introduction to finite fields and their applications*. Cambridge University Press, revised edition, 1994.

[112] Kieran F. Lim, Gleb Beliakov, and Lynn Margaret Batten. A new method for locating the global optimum: Application of the cutting angle method to molecular structure prediction. In Peter M. A. Sloot, David Abramson, Alexander V. Bogdanov, Jack Dongarra, Albert Y. Zomaya, and Yuri E. Gorbachev, editors, *International Conference on Computational Science*, volume 2660 of *Lecture Notes in Computer Science*, pages 1040–1049. Springer, 2003.

[113] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.*, 17(2):373–386, 1988.

[114] Musa A. Mammadov. A new global optimization algorithm based on dynamical systems approach. In *Proceedings 6th International Conference on Optimization: Techniques and Applications (ICOTA6)*, Ballarat, December 2004.

[115] Mitsuru Matsui. Linear cryptoanalysis method for DES cipher. In Helleseth [84], pages 386–397.

[116] Mitsuru Matsui. The first experimental cryptanalysis of the Data Encryption Standard. In Desmedt [66], pages 1–11.

[117] Mitsuru Matsui. New structure of block ciphers with provable security against differential and linear cryptanalysis. In Dieter Gollmann, editor, *FSE*, volume 1039 of *Lecture Notes in Computer Science*, pages 205–218. Springer, 1996.

[118] Mitsuru Matsui, editor. *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*. Springer, 2009.

[119] Krystian Matusiewicz, María Naya-Plasencia, Ivica Nikolic, Yu Sasaki, and Martin Schläffer. Rebound attack on the full Lane compression function. In Matsui [118], pages 106–125.

[120] Edward J. McCluskey. Minimzation of Boolean functions. *Bell Syst. Tech. Journal*, 35(5):1417–1444, 1956.

[121] Florian Mendel, Thomas Peyrin, Christian Rechberger, and Martin Schläffer. Improved cryptanalysis of the reduced grøstl compression function, ECHO permutation and AES block cipher. In Jr. et al. [95], pages 16–35.

[122] Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In Dunkelman [67], pages 260–276.

[123] Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. Rebound attacks on the reduced Grøstl hash function. In Pieprzyk [142], pages 350–365.

[124] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

[125] S. Murphy. The effectiveness of the linear hull effect. Technical Report RHUL-MA-2009-19, Royal Holloway, 2009.

[126] Marko M. Mäkelä. Survey of bundle methods for nonsmooth optimization. *Optimization Methods and Software*, 17(1):1–19, 2002.

[127] Tomislav Nad, Mario Lamberger, and Vincent Rijmen. Numerical solvers and cryptanalysis. *Journal of Mathematical Cryptology*, 3(3):249–263, 2009.

[128] Jorge Nakahara, Pouyan Sepehrdad, Bingsheng Zhang, and Meiqin Wang. Linear (hull) and algebraic cryptanalysis of the block cipher PRESENT. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS*, volume 5888 of *Lecture Notes in Computer Science*, pages 58–75. Springer, 2009.

[129] Svetla Nikova. WildPool v0.3, 2008. Personal communication.

[130] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Using normal bases for compact hardware implementations of the AES s-box. In Rafail Ostrovsky, Roberto De Prisco, and Ivan Visconti, editors, *SCN*, volume 5229 of *Lecture Notes in Computer Science*, pages 236–245. Springer, 2008.

[131] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization.* Springer-Verlag, Berlin, 2nd edition, 2006.

[132] Kaisa Nyberg. Differentially uniform mappings for cryptography. In Helleseth [84], pages 55–64.

[133] Kaisa Nyberg. Linear approximations of block ciphers. In De Santis [63], pages 439–444.

[134] Kaisa Nyberg. Correlation theorems in cryptanalysis. *Discrete Applied Mathematics*, 111(1-2):177–188, 2001.

[135] Kaisa Nyberg and Lars R. Knudsen. Provable security against a differential attack. *J. Cryptology*, 8(1):27–37, 1995.

[136] Luke O'Connor. On the distribution of characteristics in bijective mappings. In Helleseth [84], pages 360–370.

[137] Luke O'Connor. Properties of linear approximation tables. In Preneel [143], pages 131–136.

[138] Luke O'Connor. On the distribution of characteristics in bijective mappings. *J. Cryptology*, 8(2):67–86, 1995.

[139] Kenji Ohkuma. Weak keys of reduced-round PRESENT for linear cryptanalysis. In Jr. et al. [95], pages 249–265.

[140] S. Perlis. Normal bases of cyclic fields of prime-power degree. *Duke Math. J.*, 9(3):507–517, 1942.

[141] Thomas Peyrin. Improved differential attacks for ECHO and Grøstl. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 370–392. Springer, 2010.

[142] Josef Pieprzyk, editor. *Topics in Cryptology - CT-RSA 2010, The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1-5, 2010. Proceedings*, volume 5985 of *Lecture Notes in Computer Science*. Springer, 2010.

[143] Bart Preneel, editor. *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*, volume 1008 of *Lecture Notes in Computer Science*. Springer, 1995.

[144] Vincent Rijmen. *Cryptanalysis and design of iterated block ciphers*. PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium, 1997.

[145] R.L. Rivest, M.J.B. Robshaw, R. Sidney, and Y L. Yin. The RC6 block cipher. In *First Advanced Encryption Standard (AES) Conference*, page 16, 1998.

[146] Matthew J. B. Robshaw and Olivier Billet, editors. *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*. Springer, 2008.

[147] Andrea Röck and Kaisa Nyberg. Exploiting linear hull in Matsui's Algorithm 1. In *The Seventh International Workshop on Coding and Cryptography, WCC*, April 2011. to appear.

[148] Ivo G. Rosenberg. Reduction of bivalent maximization to the quadratic case. *Cahiers du Centre d'Etudes de Recherche Opérationelle*, 17:71–74, 1995.

[149] Yu Sasaki, Yang Li, Lei Wang, Kazuo Sakiyama, and Kazuo Ohta. Non-full-active super-sbox analysis: Applications to ECHO and Grøstl. In Abe [1], pages 38–55.

[150] Martin Schläffer. Subspace distinguisher for 5/8 rounds of the ECHO-256 hash function. In Biryukov et al. [25], pages 369–387.

[151] Helga Schramm and Jochem Zowe. A version of the bundle idea for minimizing a nonsmooth function: conceptual idea, convergence analysis, numerical results. *SIAM Journal of Optimization*, 2(1):121–152, 1992.

[152] Ali Aydin Selçuk. On probability of success in linear and differential cryptanalysis. *J. Cryptology*, 21(1):131–147, 2008.

[153] Ali Aydin Selçuk and Ali Biçak. On probability of success in linear and differential cryptanalysis. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN*, volume 2576 of *Lecture Notes in Computer Science*, pages 174–185. Springer, 2002.

[154] Aris Spanos. *Probability Theory and Statistical Inference: Econometric Modeling with Observational Data.* Cambridge University Press, 1999.

[155] Jacques Stern, editor. *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science.* Springer, 1999.

[156] Elmar Tischhauser. Nonsmooth cryptanalysis, with an application to the stream cipher MICKEY. *Journal of Mathematical Cryptology*, 4(4):317–348, 2011.

[157] Stefan Ulbrich. Nichtglatte Optimierung. Lecture notes, TU Darmstadt, 2005.

[158] Serge Vaudenay. Provable security for block ciphers by decorrelation. In Michel Morvan, Christoph Meinel, and Daniel Krob, editors, *STACS*, volume 1373 of *Lecture Notes in Computer Science*, pages 249–275. Springer, 1998.

[159] Serge Vaudenay. Decorrelation: A theory for block cipher security. *J. Cryptology*, 16(4):249–286, 2003.

[160] Meiqin Wang. Differential cryptanalysis of reduced-round PRESENT. In Serge Vaudenay, editor, *AFRICACRYPT*, volume 5023 of *Lecture Notes in Computer Science*, pages 40–49. Springer, 2008.

[161] Meiqin Wang, Yue Sun, Elmar Tischhauser, and Bart Preneel. A Model for Structure Attacks, with Applications to PRESENT and Serpent. In Anne Canteaut, editor, *Fast Software Encryption, FSE 2012*, Lecture Notes in Computer Science, page 20, Washington,USA, 2012. Springer-Verlag. to appear.

[162] D V. Widder. *Advanced Calculus.* Dover Publications, 2nd edition, 1989.

Curriculum vitae

Elmar Tischhauser was born in 1982 in Ludwigsburg, Germany. He studied computer science and mathematics at Technische Universität Darmstadt, Germany, receiving a Diplom-Informatiker (equivalent to M.Sc.) degree in 2007. His master thesis dealt with analysing block cipher constructions based on T-functions. In April 2008, he joined the research group COSIC (Computer Security and Industrial Cryptography) of the electrical engineering department at Katholieke Universiteit Leuven in Belgium as a Ph.D. student, again mainly working on symmetric cryptography. Since October 2008, his research was sponsored by a scholarship of the Fund for Scientific Research, Flanders (FWO–Vlaanderen).

List of publications

International Journals

1. K. Ideguchi, E. Tischhauser, B. Preneel, "Internal Differential Collision Attacks on the Reduced-Round Grøstl-0 Hash Function," *Designs, Codes and Cryptography*, 2012, *to appear.*
2. E. Tischhauser, "Nonsmooth Cryptanalysis, with an Application to the Stream Cipher MICKEY," *Journal of Mathematical Cryptology 4(4)*, pp. 317–348, 2011.
3. P.S.L.M. Barreto, V. Nikov, S. Nikova, V. Rijmen, E. Tischhauser, "Whirlwind: a New Cryptographic Hash Function," *Designs, Codes and Cryptography 56(2-3)*, pp. 141–162, 2010.

International Conferences

1. A. Bogdanov, L.R. Knudsen, G. Leander, F. Standaert, J. Steinberger, E. Tischhauser, "Key-Alternating Ciphers in a Provable Setting: Encryption Using A Small Number of Public Permutations," *Advances in Cryptology - EUROCRYPT 2012, Lecture Notes in Computer Science 7237*, T. Johansson, D. Pointcheval, Eds., Springer-Verlag, pp. 45–62, 2012.
2. M. Wang, Y. Sun, E. Tischhauser, B. Preneel, "A Model for Structure Attacks, with Applications to PRESENT and Serpent," *Fast Software Encryption, FSE 2012, Lecture Notes in Computer Science*, A. Canteaut, Ed., Springer-Verlag, 2012, *to appear.*

3. S. Schiffner, A. Pashalidis, E. Tischhauser, “On the limits of privacy in reputation systems,” *Proceedings of the 11th ACM workshop on Privacy in the electronic society (WPES 2011)*, ACM, pp. 33-42, 2011.
4. K. Ideguchi, E. Tischhauser, B. Preneel, “Improved Collision Attacks on the Reduced-Round Grøstl Hash Function,” *Information Security - 13th International Conference, ISC 2010, Lecture Notes in Computer Science 6531*, M. Burmester, S. Magliveras, G. Tsudik, Eds., Springer-Verlag, pp. 1–16, 2010.

Reports

1. S. Indesteege, E. Andreeva, C. De Cannière, O. Dunkelman, E. Käsper, S. Nikova, B. Preneel, E. Tischhauser, “The Lane Hash Function,” submission to the NIST SHA-3 competition, 72 pages, 2008.

Arenberg Doctoral School of Science, Engineering & Technology
Faculty of Engineering
Department of Electrical Engineering (ESAT)
Computer Security and Industrial Cryptography (COSIC)
Kasteelpark Arenberg 10, bus 2446
B-3001 Heverlee