KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

SHANGHAI JIAO TONG UNIVERSITY

School of Electronic Information and
Electrical Engineering

Design and Cryptanalysis of Symmetric Key Primitives

Qingju Wang

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

January 2016

Design and Cryptanalysis of Symmetric Key Primitives

Qingju WANG

Supervisory Committee:
Prof. dr. ir. Jean Berlamont, chair
Prof. dr. ir. Bart Preneel, supervisor
Prof. dr. ir. Vincent Rijmen, co-supervisor
Prof. dr. Dawu Gu, co-supervisor
(Shanghai Jiao Tong University)
Prof. dr. ir. Joos Vandewalle
Prof. dr. ir. Luc Van Eycken
Prof. dr. ir. Xuejia Lai
(Shanghai Jiao Tong University)
Prof. dr. ir. Christian Rechberger
(DTU and TUGraz)

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

January 2016

D/XXXX/XXXX/XX
ISBN XXX-XX-XXXX-XXX-X

上海交通大学
学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：_____

日　期：_____年_____月_____日

上海交通大学
学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于

保　密 □，在 _____ 年解密后适用本授权书。

不保密 □。

(请在以上方框内打 ✓)

学位论文作者签名：_____　　指导教师签名：_____

日　期：_____年 ____月 ____日　　日　期：_____年 ____月 ____日

Preface

Now it is time to write the most important part of this thesis. It is really my pleasure to thank all the people who help me realizing this PhD.

First of all, I would give my utterly gratitude to my promotor prof. Bart Preneel and prof. Vincent Rijmen, for giving me an opportunity to start a PhD in COSIC. Before moving to Belgium from China as an exchange student, I did not have any topic in my mind, not even knowledge about symmetric-key cryptography. However, with the help and encouragement from you, I became more and more interested in the area of symmetric-key cryptanalysis. Though during the last 5 years, there were many challenges, I could always get guide and advice from you before I gave up. Especially during the final writing, your insightful comments and corrections greatly helped to improve the quality of this thesis. I am indebted to Vincent's patience with my questions and tolerance towards my stupid mistakes.

I am very grateful to my promotor prof. Dawu Gu who was supporting me all the time, and encouraged me to pursuit a PhD in COSIC. Without your help, I could not finish my PhD. I am especially grateful for your traveling to Belgium attending my preliminary defence.

I would like to express thanks to other jury members: prof. Xuejia Lai, prof. Christian Rechberger, prof. Joos Vandewalle, prof. Luc Van Eycken, for their valuable feedback to this thesis. I am thankful to prof. Jean Berlamont for chairing the jury. Special thanks go to Christian, for your next travel to China for my public defence.

It is my pleasure to have the chance to collaborate with many brilliant researchers during my PhD: Hoda AlKhzaimi, Elena Andreeva, Begül Bilgin, Andrey Bogdanov, Lei Cheng, Itai Dinur, Miroslav Knežević, Chao Li, Ruilin Li, Wei Li, Yunwen Liu, Zhiqiang Liu, Atul Luykx, Bart Mennink, Willi Meier, Florian Mendel, Nicky Mouha, Yu Sasaki, Bing Sun, Yosuke Todo, Deniz Toz, Kerem Varıcı and Kan Yasuda. Many thanks go to Andrey who guided and taught me

at the very beginning of my research, I am very grateful for the enlightening discussions with you. Thanks to Nicky and Meiqin for your help in all aspects. Thanks to Kerem, Deniz and Elmar for fruitful discussions and lunch time companion. Thanks to Willi and Itai, I really learned a lot from working with you. Thanks to Begül, Bohan, Chaoyun, Tomer and Yunwen, for your precious discussions. Thanks to Atul, not only for sharing the office and translating the abstract to Dutch, also for listening to my research and giving valuable advice from another side of symmetric-key cryptography. I also thank András, Bin Zhang, Bing, Hua, Junfeng, Li, Long, Mina, Ren, Wei, Wentao and Wenyu, for your friendship and generous help.

I thank the dearest Péla for not only helping me in administrative works, but also for your time listening to me about all stuffs. I am thankful to Elsy and Wim for handling the financial issues. Because of all of you in COSIC, I had a really unforgettable life in Leuven.

I also give thanks to XDJM at Loccs, Zhiqiang, Ya, Juanru, Ning, Haihua, Yuanyuan, Junrong, Haining, Zheng, Shifeng, Chen, Bo and Bailan.

To my non-academic friends Na, Jiàn, Jiān, Huimin, Zejing, Yinghe, Zhijun, though they might never read my thesis, but I still would like to thank you.

I am deeply grateful to my families, for your forever trust and support though you do not understand what I was really doing. Finally, to Zhe, my husband, thank you so much for everything!

Qingju Wang
Leuven, March 2016

Abstract

This thesis deals with symmetric-key algorithms and more specifically block ciphers and stream ciphers. It is divided into three parts.

In the first part, we introduce the mixed integer linear programming (MILP) technique and discuss its applications in symmetric-key primitives. Differential and linear cryptanalysis are two of the most powerful statistical techniques to analyse symmetric-key primitives. For modern ciphers, resistance against these attacks is therefore a mandatory design criterion. We propose a MILP-based technique to prove security bounds against both differential and linear cryptanalysis. Our technique significantly reduces the workload of designers and cryptanalysts. MILP finds applications in this thesis in the following cases: We prove differential and linear upper bounds for the stream cipher Enocoro-128v2. For CLEFIA-type generalized Feistel networks (GFNs) with diffusion switching mechanism (DSM), we prove tighter lower bounds on the number of linearly active S-boxes, and deliver the first evidence that DSM provides an advantage by guaranteeing more active S-boxes in GFNs. Moreover, in the design of the underlying permutation of the CAESAR (Competition for Authenticated Encryption: Security, Applicability, and Robustness) candidate PRIMATEs, MILP helps to determine the offsets of the ShiftRows operations, and provides the upper bounds of the permutation against differential/linear and collision. In addition, by using MILP we construct related-key rectangle distinguishers of Rijndael-160/160 and Rijndael-192/192, based on which we can achieve the best attacks in terms of attacked rounds.

Secondly, we contribute to the cryptanalysis of block ciphers: To the best of our knowledge, our impossible-differential cryptanalysis results of Rijndael-224 and Rijndael-256 are the best in terms of round. We also analyse NSA's recent lightweight design SIMON by applying integral and zero-correlation linear cryptanalysis. As a final contribution of this part, we successfully analyse a design for MPC and FHE, which is presented at Eurocrypt 2015, and refute the designers' security claim.

Thirdly, we contribute to the research of links among statistic cryptanalysis methods by deriving the link between the impossible-differential, zero-correlation linear and integral cryptanalysis.

Abstract

In deze thesis beschouwen we symmetrische-sleutel algoritmes, meer bepaald blokcijfers en stroomcijfers. De thesis bestaat uit drie delen. In een eerste deel introduceren we de *mixed integer linear programming (MILP)* methode, en bespreken we toepassingen daarvan op symmetrische-sleutel primitieven. Differentieel en lineaire cryptanalyze zijn twee van de krachtigste statistische technieken om symmetrische-sleutel primitieven mee aan te vallen. Weerstand tegen deze aanvallen is een noodzakelijke criterium voor moderne cijfers. Een MILP-gebaseerde techniek wordt voorgesteld om beveiligings-grenzen te bewijzen voor differentieel en lineaire cryptanalyze. Door gebruik van onze techniek kan werklast voor ontwerpers en cryptanalysten sterk worden verminderd. MILP wordt in de volgende toepassingen in deze thesis gebruikt: We bewijzen differentiële en lineaire bovengrenzen voor het stroomcijfer Enocoro-128v2. Voor CLEFIA-type gegeneraliseerde Feistel netwerken (GFNs) met diffusie-wisselings mechanismen (DSM) bewijzen we scherpere ondergrenzen op het aantal lineair-actieve S-boxen, en leveren we het eerste bewijs dat DSM meer actieve S-boxen garandeert in GFNs. Verder nog, in het ontwerp van de onderliggende permutaties van de CAESAR (*Competition for Authenticated Encryption: Security, Applicability, and Robustness*) kandidaat PRIMATEs, helpt MILP om de *offsets* van de ShiftRows operaties te bepalen, en geeft bovengrenzen op weerstand tegen differentieel/lineair analyze en botsingen voor de permutatie. Daarnaast wordt een gerelateerde-sleutel, rechthoekige onderscheider van Rijndael-160/160 en Rijndael-192/192 geconstrueerd via het gebruik van MILP, waardoor de beste aanvallen in termen van aantal rondes bereikt worden.

In een tweede deel bekijken we de cryptanalyze van blokcijfers: voor zover wij weten zijn onze onmogelijke-differentieel cryptanalyze resultaten van Rijndael-224 en Rijndael-256 de beste in termen van rondes. SIMON, een recent lichtgewicht ontwerp van de NSA, wordt ook geanalyseerd via een toepassing van integraal en nul-correlatie lineaire cryptanalyze. Als laatste contributie van dit deel wordt een beveiligings-stelling van een ontwerp voor MPC en FHE van

Eurocrypt 2015 weergelegd.

Ten laatste worden verbanden tussen twee statistische cryptanalytische methodes onderzocht, namelijk onmogelijke-differentieel -en integraal-cryptanalyze.

摘要

分组密码作为密码学中重要的组成部分，在许多密码算法的构造中起到了重要作用。本论文主要阐述了对称密码算法，尤其是分组密码和流密码算法的设计与安全性分析方法。论文共分为三个部分。

首先，我们介绍了混合整数线性规划（MILP），并讨论了其在对称密码安全性分析中的应用。众所周知，差分分析和线性分析方法是对称密码算法分析最有效的两种统计分析方法。所以抵抗差分分析和线性分析是现代密码算法设计的最重要原则。我们提出一种基于 MILP 的方法，可以给出算法针对差分分析和线性分析的安全界。我们的方法应用范围广泛，能够大大减少算法设计者和分析人员的工作量，进而提高其工作效率。本论文将 MILP 成功地应用在对称密码算法的几个方面：首先我们成功证明了流密码 Enocoro-128v2 的差分和线性分析的安全上界。其次，对于具有扩散转换机制（DSM）的类 CLEFIA 的一般 Feistel 结构（GFN），我们证明了更好的线性活性 S 盒的下界，并首次证明了 GFN 采用 DSM 的扩散方法，可以获得更多的活性 S 盒，进而提高算法抵抗线性分析的能力。再次，在认证加密算法 CAESAR 竞赛候选算法 PRIMATEs 的设计中，基于抵抗差分分析和线性分析的理论，我们利用 MILP 的方法搜索选定底层置换 PRIMATE 的行变换（ShiftRows）的值，并且给出 PRIMATE 抵抗差分分析、线性分析和碰撞攻击的安全上界。最后，利用 MILP 的方法，我们首先构造了 Rijndael-160/160 和 Rijndael-192/192 的相关密钥矩形区分器，并在此基础上对其分别给出了到论文写作时最多轮数的攻击。

其次，本论文也研究了分组密码的安全性分析：首先我们对 Rijndael-224 和 Rijndael-256 进行不可能差分分析，据我们所知，我们的不可能差分攻击仍是最好的攻击结果。其次我们也对 NSA 设计的轻量级分组密码 SIMON 进行了分析，并首次尝试了积分攻击和零相关线性攻击。最后我们研究了 Eurocrypt2015 上提出的针对多方计算（MPC）和同态加密（FHE）设计的分组密码 LowMC，基于高阶差分区分器，我们对全轮 LowMC 做的优化插值攻击成功驳斥了设计者的安全性声明。

最后，我们研究了统计分析方法之间的联系，并建立了不可能差分分析、零相关线性分析和积分攻击之间的关联。

Abbreviations

AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
AK	AddRoundKey
ARX	Addition Rotation and XOR
CAESAR	Competition for Authenticated Encryption: Security, Applicability, and Robustness
CBC	Cipher Block Chaining
CC	Chosen-Ciphertext
CFB	Ciphertext Feed Back
CO	Ciphertext-Only
CP	Chosen-Plaintext
CTR	Counter
DP	Differential Probability
DSM	Diffusion Switching Mechanism
ECB	Electronic Code Book
EDP	Expected Differential Probability
ELP	Expected Linear Probability
FHE	Fully Homomorphic Encryption
FIPS	U.S. Federal Information Processing Standard
GCM	Galois/Counter Mode
GFN	Generalised Feistel Network
ID	Impossible-Differential Cryptanalysis
KP	Known-Plaintext

LP	Linear Probability
MAC	Message Authentication Code
MC	MixColumns
MDP	Maximum Differential Probability
MDS	Maximum Distance Separable
MEDP	Maximum Expected Differential Probability
MELP	Maximum Expected Linear Probability
MitM	Meet-in-the-Middle
MLP	Maximum Linear Probability
MPC	Multi-Party Computation
NIST	National Institute of Standards and Technology
NSA	National Security Agency
RAM	Random Access Memory
RK	Related-Key
SAT	Boolean Satisfiability Problem
SB	SubBytes
SD	Substitution-diffusion
SPN	Substitution-Permutation Network
SR	ShiftRows
WLANs	Wireless Local Area Networks
XOR	Exclusive OR
ZC	Zero-Correlation Linear Cryptanalysis

Contents

Abstract iii

Abstract v

Contents xi

List of Figures xv

List of Tables xvii

I Symmetric Key Primitives 1

1 Introduction 3

1.1 Cryptography model 3

1.2 Symmetric key primitives 5

1.2.1 Block ciphers 5

1.2.2 Stream ciphers 6

1.2.3 Hash functions 7

1.2.4 Message authentication codes 8

1.2.5 Authenticated encryption 9

1.3 About this dissertation 10

2 Block Ciphers **11**

2.1 Definitions 11

2.2 Design methods 12

2.2.1 Iterated design method 13

2.2.2 Substitution-Permutation networks 14

2.2.3 Feistel networks 16

2.2.4 Lai-Massey schemes 20

2.3 Modes of operation 21

2.4 Conclusions 22

3 Cryptanalysis Methods **23**

3.1 Attack models 23

3.1.1 Attack data types 23

3.1.2 Attack key settings 24

3.1.3 Attack goals 25

3.1.4 Attack complexity metrics 26

3.2 Differential cryptanalysis and extensions 27

3.2.1 Differential cryptanalysis 27

3.2.2 Impossible-differential cryptanalysis 29

3.2.3 Boomerang attack 29

3.2.4 Rectangle attack 31

3.2.5 Higher-order differential attack 32

3.2.6 Integral attack 34

3.2.7 Local collision 34

3.3 Linear cryptanalysis and extensions 35

3.3.1 Linear cryptanalysis 35

3.3.2 Zero-correlation linear cryptanalysis 37

3.4 Algebraic cryptanalysis 38
3.4.1 Interpolation attack 38
3.5 Conclusions 39

4 Contributions of this thesis **41**
4.1 MILP and its applications to symmetric-key primitives 41
4.1.1 Backgrounds 42
4.1.2 Applications to stream cipher Enocoro-128v2 42
4.1.3 Applications to AES 43
4.1.4 Applications to GFN$_d$-II 43
4.1.5 Design PRIMATEs permutation 43
4.1.6 Applications to Rijndael in the related-key model 44
4.2 ID, ZC, integral cryptanalysis and their links 44
4.2.1 Impossible-differential attacks on Rijndael 44
4.2.2 Integral and zero-correlation attack on SIMON 44
4.2.3 The links among ID, ZC and integral cryptanalysis . . 46
4.3 Interpolation attack of LowMC 46
4.4 Conclusions 47

5 Conclusion and future work **49**
5.1 MILP and its applications 49
5.2 Cryptanalysis of block ciphers 50
5.3 Links among cryptanalysis methods 51
5.4 Directions for future work 51

Bibliography **53**

II Publications 71

6 Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming 75

7 The provable constructive effect of diffusion switching mechanism in CLEFIA-type block ciphers 97

8 Related-Key Rectangle Cryptanalysis of Rijndael-160 and Rijndael-192 111

9 PRIMATEs 139

10 Improved Impossible Differential Attacks on Large-Block Rijndael 165

11 Cryptanalysis of Reduced-Round SIMON32 and SIMON48 183

12 Links Among Impossible Differential, Integral and Zero Correlation Linear Cryptanalysis 203

13 Optimized Interpolation Attacks on LowMC 225

A 253

Curriculum Vitae 255

List of Figures

1.1 Model of a two-party communication using encryption 4

2.1 Key alternating block cipher construction 13
2.2 One round of a Substitution-Permutation network 15
2.3 One round of a Feistel network 17
2.4 Generalised and unbalanced Feistel networks 18
2.5 One round of Feistel networks with Substitution-Permutation . 18
2.6 SPECK round function 19
2.7 One round Lai-Massey scheme 21

3.1 Impossible-differential cryptanalysis 30
3.2 A right quartet of the boomerang distinguisher 31
3.3 A local collision in AES-256 35

List of Tables

2.1 Specific examples of Feistel structures 19

Part I

Symmetric Key Primitives

Chapter 1

Introduction

Cryptography can be dated back to over 4000 years ago, but widespread use began with the transmission of data via radio channels in the early 1900s. For most cryptographers, modern cryptographic theory starts with Shannon in the 1940s [147, 148]. Hence, cryptology can also be considered as a very young research field. Nowadays, cryptography is a well established area of computer science. A wide range of applications of cryptography allow us to transfer confidential data over insecure communication channels, execute bank transactions online, and sign contracts over the internet, etc. These applications are based on cryptographic primitives that provide valuable services such as encrypting messages and authenticating entities.

1.1 Cryptography model

The primary goal cryptography aims at is to keep messages confidential. We start describing cryptography with simple two-party model. The two parties, the message *sender* and *receiver* of the communication, are typically named Alice and Bob. They first choose a key pair (e, d) which is only shared by themselves. When Alice wants to send a *plaintext* m to Bob, she uses the secret key e to compute the *ciphertext* $c = E_e(m)$ (this operation is called *encryption*), and transmits it to Bob over an insecured channel. After receiving c, Bob uses the secret key d he shared with Alice to compute $E_d(c) = m$ (this operation is called *decryption*), and recovers the original message m. The malicious adversary Eve tries to defeat the secure communication goal of Alice and Bob. Even if she is capable of eavesdropping, disrupting and modifying the data on the channel,

she does not know the secret key d, and she cannot decrypt the ciphertext and read the original message. The model of a two-party communication is depicted in Figure 1.1.

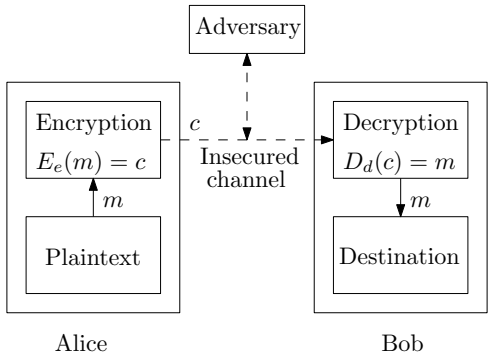


Figure 1.1: Model of a two-party communication using encryption

Most people believe that cryptography is merely keeping communications secret. However, cryptography attains the following four main goals:

Confidentiality: Being the original purpose of cryptography, confidentiality ensures that the message sent by Alice can only be understood by the authorized receiver Bob. Though Eve can eavesdrop the ciphertext from the channel, she cannot infer the message exchanged between the communication parties.

Data Authentication: When Eve can modify the data being transferred over the insecure channel, the communicating parties must be able to verify the integrity of the data. *Hash functions* and *message authentication codes* are typically used to achieve this goal.

Entity/Sender Authentication: This property assures the user communicate with the intended parties. In particular, it is not allowed for Eve to send a message to Bob, appearing as if it originates from Alice, or vice versa. Entity authentication is usually achieved by using identity cards, biometric data, login passwords, etc.

Non-Repudiation (of origin or receipt): It means that the communicating parties will not be able to deny that they have sent or received a message. It is mostly achieved by verifying digital signatures.

Not all four goals are required in a single real-world application. Their importance can vary depending on the application. For example, while confidentiality is more important for communication, authenticity is crucial for financial transactions.

The cryptography constructions that are trying to achieve one or more of the above properties, can be classified into three categories according to the manner the key pairs (e, d) are using: symmetric, asymmetric and unkeyed primitives.

In an symmetric-key system, as the name suggested, the encryption key e and the decryption key d are the same. This exact secret key must be agreed by both the communication parties preceding a communication. These primitives are also referred to as secret-key primitives.

Unlike in symmetric key systems, in an asymmetric-key encryption two different keys are used. One key e is publicly distributed to everyone who wishes to communicate with Bob, while the other key d is only held by Bob. This is where the other name public-key encryption comes from.

Symmetric-key systems and public-key encryption schemes have a number of complementary advantages. In particular it is commonly accepted (though there is no proof showing that this must be the case) that the computational performance of public-key encryption is inferior to that of symmetric-key system. Modern cryptographic systems take advantage of the strengths of each by applying public-key encryption to establish a key for a symmetric-key system. This obliges the communication parties to simultaneously enjoy the long term nature of the key pairs (e, d) and the performance efficiency of the symmetric-key system.

Finally, the category of unkeyed primitives are constructions, as the name indicates, require neither a shared secret key nor a public key to serve their purpose. One of the examples for unkeyed primitives is cryptographic *hash functions.*

In this thesis, we focus on the cryptanalysis and design of symmetric-key primitives, especially block ciphers.

1.2 Symmetric key primitives

1.2.1 Block ciphers

Block ciphers form a subset of symmetric-key ciphers; their main goal is to provide confidentiality. Block ciphers play an important role as building block

in the design of many other cryptographic primitives, including stream ciphers, hash functions, MAC algorithms and Authenticated Encryption.

Important examples of block ciphers are DES (Data Encryption Standard) and 3DES. DES was proposed in the 1970s, and became one of the most widely used symmetric-key ciphers. Since its introduction, DES has drawn a substantial amount of analysis [29, 115]. Because of the 56-bit key size, the increasing computational power made brute-force attacks feasible. 3DES, which applies DES three times, is still used since it can provide more security than DES and for legacy reasons (3DES can be implemented on a DES hardware platform). However, 3DES is slow and the 64-bit block length limits the security level it can achieve. Therefore, since the end of 1980s, alternative block cipher designs were introduced. Some examples of such designs are FEAL [123], IDEA [103], LOKI [49], Blowfish [143], SAFER [113], RC5 [137] and CAST [7].

AES competition

In 1998, the National Institute of Standards and Technology (NIST) announced a competition to select the Advanced Encryption Standard (AES) to replace DES. After three years, five finalist MARS [50], RC6 [138], Rijndael [58], Serpent [23] and Twofish [145] remained from the 15 candidates. A subset of Rijndael with a block size of 128 bits and three different key sizes (128, 192 and 256 bits), was chosen as the AES algorithm in 2001. Up to now, there have been no practical attacks to AES except side channel attacks which actually are implementation attacks rather than attacks on the algorithm itself.

1.2.2 Stream ciphers

Stream ciphers form a subset of symmetric-key primitives. Compared to block ciphers, stream ciphers operate on smaller units - typically bits or bytes. Inspired by the one-time pad, stream ciphers act as pseudo-random bit generators. They generate a pseudo-random sequence called the *key stream* from the secret key and an initial value IV, which is mixed with the plaintext to obtain the ciphertext; the typical mixing operation is XOR. The same key stream is used for the decryption.

Stream ciphers are often used for their speed and simplicity of implementation in hardware, in particular where the length of plaintext is variable or unknown in advance. If block cipher were used in this case, either efficiency or implementation complexity would have to be chosen, since block ciphers cannot directly work on blocks shorter than their block sizes and a padding algorithm

might have to be deployed. On the contrary, stream ciphers can avoid this problem by operating on the much smaller units (usually bytes) that can be transmitted.

We can classify stream ciphers into two types: *synchronous* and *self-synchronizing*. In a self-synchronizing stream cipher, the key stream is generated using the secret key, the IV and some of previous ciphertext bits. By contrast, in a synchronous stream cipher, the key stream is generated only using the secret key and the IV of the cipher, and thus is independent of the plaintext. Synchronous stream ciphers are widely used, examples include RC4 [135] – the most widely used stream cipher in software, and the eSTREAM finalists, e.g. Trivium [64]. Enocoro-128v2 [169] designed by Hitachi in 2010 is another example.

eSTREAM competition

Realising the need for better stream ciphers, community made an attempt in 2004 to resuscitate stream ciphers through an open contest called eSTREAM [86]. The original call for proposals generated considerable interest with 34 proposals submitted. The project ended in April 2008 with an announcement of two profiles containing eight stream ciphers, where four stream ciphers more suitable for software applications with high throughput requirements belong to software-oriented profile and the other four stream ciphers particularly suitable for hardware applications with restricted resources belong to the hardware-oriented profile. An update contains three stream ciphers in hardware profile was published in September 2008 [73].

1.2.3 Hash functions

A cryptographic hash function is a function that maps arbitrary length bit strings to fixed length outputs. Cryptographic hash functions were first proposed by Diffie and Hellman [66] who used them to construct more efficient digital signatures. Ever since, the use of hash functions was extended to various applications, including password protection, key generation, entropy extraction and message authentication, each of which demands different security properties.

Following Merkle's security model [121], the main properties a hash function should satisfy are collision resistance, second preimage resistance, and preimage resistance. The traditional way of designing a hash function is by starting with a compression function that only allows fixed length inputs, next a mode of operations are designed to obtain a hash function that accommodates arbitrarily

long inputs. Therefore, designing a secure hash function was shifted to designing a secure compression function f. Two main approaches to do this are: to build f from one or more block ciphers or permutations.

The block cipher approach can be dated back to Rabin's compression function in 1978. This approach was generalized by Matyas, Meyer and Oseas in [117], Davies and Meyer in [61], and Miyaguchi and Preneel in [132] and [122] independently. Prominent examples that use this approach include MD4 [136] and its strengthened version MD5 [134], SHA-0/1/2, Whirlpool [14], and many others. In 1996, Dobbertin found collisions for MD4 by using differential cryptanalysis [70]. Later, Wang et al. improved this result and presented collisions on MD4 [167], MD5 [168] and SHA-0 [168] by applying message modification techniques. By further exploiting these techniques, hash functions deploying the similar design ideas such as SHA-1 [63], SHA-2 [119], SM3 [120], HAS-160 [118] were analysed.

The second design approach is based on permutations; it has obtained more attention recently with the introduction of the sponge methodology by Bertoni et al. [19], with Keccak/SHA-3 [20] as its most prominent example. Note that the compression function itself is not secure.

SHA-3 competition

In response to the reduction of confidence in MD4, MD5 and SHA-2, NIST announced in 2007 a competition to choose a new hash function standard: Secure Hash Algorithm 3 (SHA-3) [126]. The competition received 64 submissions from all over the world in 2008. NIST selected 51 of them to round one, and fourteen of them to round two. In December 2010, five candidates were announced to the final round of the competition: BLAKE [13], Grøstl [77], JH [171], Keccak [20] and Skein [75]. Among these five algorithms, KECCAK [20] was chosen as the new hash standard in October 2012.

1.2.4 Message authentication codes

A Message Authentication Code (MAC) algorithm can be considered as a keyed fingerprint computed on a plaintext of any length. The purpose of the MAC is to provide both integrity and authenticity. A MAC algorithm consists of three steps: *Key generation*, *MAC generation* and *MAC verification*. In a MAC algorithm, Alice and Bob agree on a secret key K in advance as in a symmetric-key system. In the MAC generation algorithm, Alice computes from the message m and the secret key K a MAC value $\mathrm{MAC}_K(m)$, and appends

it to the plaintext when sending to Bob. In MAC verification algorithm, Bob uses the shared secret key K to compute the MAC value $\text{MAC}_K(m)$ himself, and verifies whether it matches the one he received. If so, he believes that the message indeed came from Alice. Otherwise, a modification might have happened to the plaintext or the MAC value.

The informal security state of a MAC is that it should be computationally unfeasible to generate a modified plaintext together with a modified MAC that appears valid to Bob. Widely used MACs include CBC-MAC, hash-based MAC (HMAC) and parallelizable MAC (PMAC).

1.2.5 Authenticated encryption

If one wants to achieve confidentiality and integrity simultaneously from one primitive, authenticated encryption (AE) is exactly what we need. There are requirements from the application setting where we wish to not only encrypt and authenticate messages, but to include auxiliary data which should be authenticated, but left unencrypted. Hence the associated data should be included as input to the AE schemes, we call them as authenticated encryption with associated data (AEAD). There are two approaches to construct AE: generic composition and dedicated scheme.

In a generic composition approach, authentication encryption can be obtained by combining two separate primitives with two independent keys, one for encryption (typically provided by block ciphers), the other one for authentication (typically provided by a MAC algorithm). There are three obvious choices: *MAC-then-Encrypt* (MtE), *Encrypt-then-MAC* (EtM), and *Encrypt-and-MAC* (E&M). Among these, EtM with a secure encryption scheme and a secure MAC each with independent keys is the best approach for achieving AE. The cost of such an algorithm is much higher than any one of the primitives it originated from because of the two independent keys required.

The dedicated AE schemes are mainly constructed based on block ciphers and permutations. Because only one key is used to achieve both the encryption and the authentication, they have more advantages in efficy than the generic composition approach. Important block cipher based examples include IAPM [90], OCB [100,139,140], XECB [79], CCM [72], GCM [71], SIV [141], BTM [85] and McOE-G [76]. Permutation based approach, such as SpongeWrap [21] are obtaining attention recently.

The two main designs of AEAD recommended by NIST are Galois/Counter Mode (GCM) [71] and Counter with CBC-MAC (CCM) [72], both are block cipher modes of operations. GCM adopting AES as the underlying primitives

(denoted as GCM-AES), is part of the TLS 1.2 cipher suite [65]. However, due to the weak key problem that has been pointed out by Furguson [74] in 2005, Joux [89] in 2006, Handschuh and Preneel [82] in 2008, Saarinen [142] in 2012 and Procter and Cid [133] in 2013, the trustworthiness of GCM-AES is reduced.

CAESAR competition

In response to the current status of GCM-AES, and the increasing requirements in various applications from the cryptographic community, a competition for authenticated encryption: security, applicability, and robustness (CAESAR) [18] was initiated by Bernstein in 2012 to identify a new portfolio of AEAD schemes. 57 submissions in total were received in the first round of the competition; 29 of them went to the second round, our design PRIMATEs [11] based on permutation is one of them. The tentative announcement of the final portfolio was set to the end of 2017.

1.3 About this dissertation

This dissertation is based on publications, and consists of two parts. The first part gives an introduction to symmetric-key cryptography, more precisely to the relevant background information about block ciphers. It also provides a brief outline of our contributions to the design and analysis of block ciphers. The second part consist of our publications.

This first part consists of five chapters. Chapter 1 (this chapter), introduces the field of symmetric-key cryptography. Chapter 2 focuses on block ciphers; it discusses the definitions, the design methodologies and the modes of operation. Chapter 3 explains the important cryptanalysis methods we will apply in this thesis, and Chapter 4 gives a summary of our contributions to the analysis and design of block ciphers. The main purpose of these chapters is to show our publications fit together. Finally, Chapter 5 concludes and discusses some possible directions for future work.

Chapter 2

Block Ciphers

Among the symmetric primitives, block ciphers are the most widely used building blocks in the construction of many other cryptographic primitives, for instance stream ciphers, hash functions, pseudo-random generators, message authentication codes and authenticated encryption. In terms of the main goals of cryptography described in Chapter 1, block ciphers aim at providing confidentiality. In Section 2.2 the design methodologies followed by modern block cipher design are described. In Section 2.3 we briefly explain the applications of block cipher modes of operations.

2.1 Definitions

A block cipher maps an input of fixed-length, called a block, together with a key, to an output block of identical length. For a fixed key, the mapping is bijective. The formal definition of block ciphers is given below.

Definition 1. *A mapping $E : \mathbb{F}_2^n \times \mathbb{F}_2^k \to \mathbb{F}_2^n$ is called a block cipher with the block size n and the key size k, if the mapping $E(\cdot, k)$ is bijective for each fixed $k \in \mathbb{F}_2^k$, that is, if the inverse mapping E^{-1} exists and $E^{-1}(E(m, k), k) = m$ for any $m \in \mathbb{F}_2^n$.*

The input m and output c of E are called the *plaintext* and the *ciphertext* respectively, k is called the key. $E(, k)$ is called the *encryption* function, and its inverse denoted by $D = E^{-1}$ is called the *decryption* function. Hence, for any fixed key $k \in \mathbb{F}_2^k$ the encryption and the decryption functions can be written as $E(m, k) = c$ and $E^{-1}(c, k) = m$, respectively.

One should know that no block cipher can serve all the cryptographic goals. The design choices made for a block cipher influence not only the security, but also efficiency and other platform-dependent characteristics. One important flavour of block ciphers is lightweight block ciphers that are designed to be efficient for limited resource environments. During recent years, many lightweight ciphers have been designed. Prominent examples include: ICEBERG [154], mCrypton [108], HIGHT [84], PRESENT [42], KATAN [62], LED [81], Piccolo [152], KLEIN [80], EPCBC [174], PRINCE [45] and TWINE [157]. In 2013, NSA also proposed two families of highly-optimized block ciphers, SIMON and SPECK [15], which can provide excellent performance in hardware respectively software implementation. In the end, the design goal of a block cipher is to offer a reasonable trade-off between security and performance.

2.2 Design methods

In 1949, Shannon introduced the twin principles of *confusion* and *diffusion* for the block cipher design in his landmark paper *Communication Theory of Secrecy Systems* [148]. They are still the most widely used principles in modern block ciphers design. They can be summarised as follows:

> **Confusion** aims to change the output bit value when the input bit value is changed in such a way that it is too complicated to determine the relation. To accomplish confusion, one can translate the input through a non-linear table created from the secret key.
>
> **Diffusion** is used to mix the input values and its goal is that changing one bit of the input will influence many bits of the ciphertext. Diffusion can be achieved in several ways. One is to divide the input into small blocks (called words) and then transform them. Another way is to treat the input as an array and shuffle the positions of the bits.

Confusion itself may let some potential patterns go through to the output, as a result the adversary may obtain more knowledge about the cipher than being expected. However, a good diffusion can help to scatter the patterns widely throughout the output of the cipher. This makes the potential patterns scramble each other and there will be no useful information for the adversary, or increases the mount of data required by an attack to such a large number that it is infeasible to achieve.

We proceed with the construction methods of block ciphers. Since modern block ciphers follow the iterated design strategy, we limit the thesis to this approach.

2.2.1 Iterated design method

Block ciphers are often constructed as iterated mappings based on a bijective round function. Let r denote the number of rounds, and the *key scheduling* algorithm expands the encryption key k into r *round keys* $(k_1, \ldots, k_r)$. Given the plaintext $m = m_0$, the ciphertext c is then obtained as $c = m_r$ with

$$m_i = F(m_{i-1}, k_i) \quad (1 \leq i \leq r) \ ,$$

where $F(\cdot, k_i)$ is the *round function* and k_i is the *round key* in the ith round respectively. One type of iterated block ciphers is referred to as key-alternating block ciphers. In such designs, the round keys are added (typically use the XOR operation) to the intermediate states in between the round functions. An r-round key-alternating block cipher is depicted in Figure 2.1. In most cases,

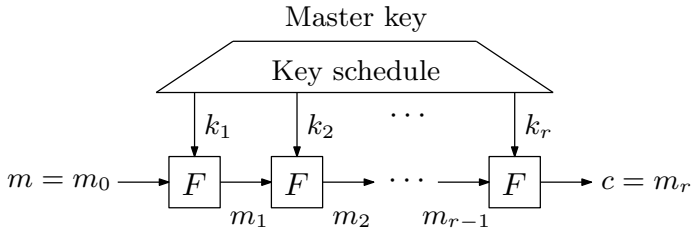


Figure 2.1: Key alternating block cipher construction

one extra round key is added before the first round or after the last round, which we do not indicate in Figure 2.1. Such a key is called a *whitening key*. It helps to frustrate an adversary's attempt to control the input to the first and/or the output from last round, based on which a meet-in-the middle (MitM) attack might be mounted.

The iterated designs strategy for block cipher has two advantages. First, as the round functions are identical (except the round keys are added in reverse order), only a slight implementation for a round is necessary in both software and hardware implementations; this greatly benefits the efficiency of the implementations.

Second, it is much easier to understand a slightly core component rather than a composition of several components. This yields obvious benefits to the designer because if the designer can understand how the block cipher behaves after iterating several times of the round function, he can choose an appropriate number of rounds for the cipher. On the other hand, the systematic design approach also enables the adversary to obtain some property of the core

component. Then she/he may extend it to a larger number of rounds, based on which she/he may break the whole block cipher.

In iterated block cipher designs, there are three main constructions: Substitution-Permutation networks (SPNs), Feistel networks and Lai-Massey schemes. In the following we first describe them; then we will describe examples that are investigated in later chapters of this thesis.

2.2.2 Substitution-Permutation networks

Substitution-Permutation Networks (SPNs) use a round function that is a combination of substitution and permutation operations. These operations are often used to confuse and diffuse the data, respectively.

For SPNs, the plaintext $m = X_0$ is processed for r rounds to obtain the corresponding ciphertext $c = X_r$. The round function $F : \mathbb{F}_2^n \times \mathbb{F}_2^k \to \mathbb{F}_2^n$ uses the n-bit intermediate state value X_{i-1} and the k-bit round-key value k_i as inputs to obtain the next intermediate state value X_i. The encryption operation can be described as:

$$X_i = F(X_{i-1}, k_i) \ ,$$

where $1 \leq i \leq r$. The round function F usually consists of three steps:

1. Substitution (S): The current state (or parts of it) values are substituted with new values in a nonlinear way. This layer is usually accomplished by using substitution boxes (S-boxes) in parallel. If each S-box operates on s out of n bits ($s \ll n$), there are n/s s-bit S-boxes working in parallel.

2. Permutation (P): The permutation is typically accomplished by using one of the two steps or both. First, all the state bits are permuted by mapping to new positions, or by dividing the state into small words and mapping them to new positions. Second, a linear mixing is applied to the state: this is typically accomplished by multiplying with a binary matrix, or with a matrix over a finite field of size 2^w, where w is the word size.

3. Key addition: In the ith round, the round key k_i is mixed with the current state. The most widely used operation is XOR.

One round of the SPN structures is illustrated in Figure 2.2. For decryption, the inverse of the F function is applied to the ciphertext with round keys in the reverse order.

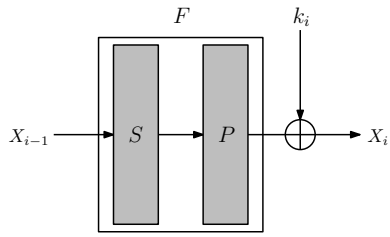


Figure 2.2: One round of a Substitution-Permutation network

AES/Rijndael The most prominent example of an SPN is the AES [60]. The design of the AES is of particular interest, as it proposed a new design approach of obtaining confusion and diffusion for a block cipher, called the wide-trail strategy [59]. Today many block ciphers follow this design strategy and we call them *AES-like* structures, examples include PRESENT [42], Threefish [75], LED [81] and LBlock [173].

In AES/Rijndael, each data block (plaintext, ciphertext, subkey or intermediate value) is represented by a $4 \times N_b$ `state matrix` of bytes, where N_b is the block size divided by 32. The state is then transformed by iterating a round function composed of the following four steps:

- `SubBytes (SB)`: a non-linear substitution (8-bit S-box) that acts on every byte of the state independently.
- `ShiftRows (SR)`: a cyclic shift of bytes in a row that acts individually on each of the rows of the state. The shift offset of each row depends on the block length N_b.
- `MixColumns (MC)`: a linear transformation (based on an MDS code over $GF(2^8)$) that acts independently on every column of the state.
- `AddRoundKey (AK)`: the xor of the round key with the intermediate state.

The steps `SubBytes` and `AddRoundKey` serve the goal of confusion, and `ShiftRows` and `MixColumns` serve the goal of diffusion.

Rijndael-b/k that have identical block size b and key size k are also denoted as Rijndael-b. We discuss the security of Rijndael-160 and Rijndael-192 against the related-key rectangle attack in Section 4.1.6 and Chapter 8. The impossible-differential attacks on Rijndael-224 and Rijndael-256 are explored in Section 4.2.1 and Chapter 11.

Other specific SPNs in this thesis

Except Rijndael, we discuss the following SPN structures in this thesis: PRIMATEs [11] is one of the second round candidates in the CAESAR competition. The underlying permutation of PRIMATEs which is called PRIMATE is inspired by Rijndael and the dedicated lightweight AE design Fides [30]. It has two different sizes, denoted as PRIMATE-80 for the 200-bit permutation and PRIMATE-120 for the 280-bit permutation, which operate on a 5×8 and a 7×8 state of 5-bit elements, respectively. PRIMATE updates the internal state by means of the sequence of transformations

$$\mathsf{CA} \circ \mathsf{MC} \circ \mathsf{SR} \circ \mathsf{SE} \ .$$

SubElements, the only non-linear operation of PRIMATE is a permutation consisting of a 5-bit S-box applied to each element of the state, where the S-box is an almost bent permutation (see Appendix A) and the maximum differential and linear probability is 2^{-4}. ShiftRows is an element transposition that cyclically shifts left the rows of the state over different offsets $s_i = \{0, 1, 2, 4, 7\}$ positions for PRIMATE-80 and $s_i = \{0, 1, 2, 3, 4, 5, 7\}$ positions for PRIMATE-120 in row i. The MixColumns step is a left multiplication by a 5×5 (resp. 7×7) matrix. We use a recursive approach to generate a Maximum Distance Separable (MDS) matrix that has a maximum branch number (6 and 8 respectively). ConstantAdditionstep XORs a predefined constant to the second element of the second row with by a bitwise XOR operation. For detailed design we refer to Section 4.1.5 and Chapter 9.

LowMC [10] is a block cipher published at Eurocrypt 2015 for Multi-party Computation (MPC) and Fully Homomorphic Encryption (FHE). The round function first employs a partial S-box layer having fewer calls to a 3-bit S-box during each round. Following the S-box layer is the permutation layer, where the state is multiplied with a binary $n \times n$ (n is the block size) matrix that is chosen independently and uniformly at random from all invertible binary $n \times n$ matrices. We explore the security of LowMC in Section 4.3 and Chapter 13.

2.2.3 Feistel networks

Feistel networks are defined as follows: Let n (even) be the block size, and r the number of rounds. Assume that the input of the round function F is divided into two branches L_0 and R_0, each of $n/2$ bits. The round function F processes

data in the ith round as

$$R_i = L_{i-1} \ ,$$

$$L_i = R_{i-1} \oplus f(L_{i-1}, k_i) \ ,$$

where $1 \leq i \leq r$. Here the *feistel function* f can be any function (not necessarily bijective) taking an $n/2$-bit input and a round key k_i and producing an $n/2$-bit output. Figure 2.3 illustrates one round of a Feistel network. Note that there is

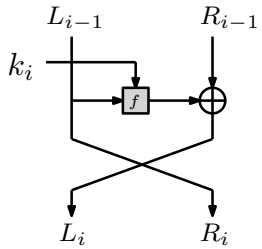


Figure 2.3: One round of a Feistel network

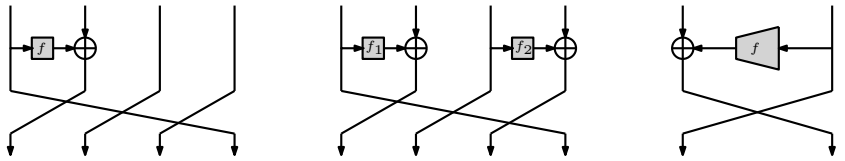
no swap operation in the last round, and the ciphertext is the concatenation of R_r and L_r, i.e. $(R_r||L_r)$. The reason behind this choice is to make the encryption and the decryption function exactly the same except for the order of the subkeys. This property reduces the size of implementations and makes Feistel networks very appealing for hardware implementation.

Variants of Feistel structures

Two variants are suggested to Feistel networks: one divides the input value into more than two (even) equal branches [129]. This variant is called Type-i generalised Feistel network GFN_j-i, where i denotes the number of f functions used in each round and j denotes the number of splitting branches of the input. The second variant divides the input into two branches with unequal sizes which is studied in [144]. One of the examples for unbalanced Feistel networks is Skipjack [127] developed by NSA. Figure 2.4 depicts the details of GFN_4-I, GFN_4-II and unbalanced GFN_2 in sequence.

Feistel networks with Substitution-Permutation round function

Another way to classify the Feistel variants is the design of the round functions. There are many ways to design a round function that is iterated for Feistel



Type-I Feistel **Type-II Feistel** **Unbalanced Feistel**

Figure 2.4: Generalised and unbalanced Feistel networks

networks. The most widely used approach is to adopt Substitution-Permutation structure as the feistel function. Figure 2.5 gives one round of the Feistel networks having Substitution-Permutation as the feistel function.

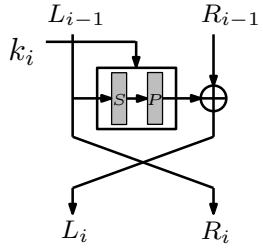


Figure 2.5: One round of Feistel networks with Substitution-Permutation

There are a large number of ciphers fitting this category. The most important example is the Data Encryption Standard (DES) [29], developed by IBM and NSA in 1976. In this thesis we discuss the security of other Feistel networks with Substitution-Permutation as the feistel function: CAST-256 [7], CLEFIA [153], Camellia [12], and SMS4 [2, 67].

Feistel-like structures with ARX

Another approach to design block ciphers is to follow the (generalised) Feistel structures partially. The round function does not strictly follow the Feistel structure; it contains three operations Addition, Rotation and XOR (ARX), where the non-linear Addition operation is very efficient on software. We call this structure the Feistel-like structure with ARX. This approach is used to design many algorithms such as RC4 [135], TEA [170], HIGHT [84] and SPECK [15]. Among these, SPECK designed by NSA is a prominent example. We give the

round function of SPECK in Figure 2.6, where $\boxplus$ is mod $2^{n/2}$ with n the block size, $\lll$ and $\ggg$ are the left and right circular shifts with the offset values of a and b determined by the block size.

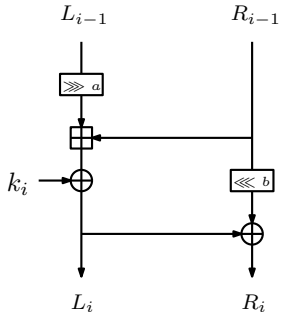


Figure 2.6: SPECK round function

Specific Feistel structures in this thesis

In this part we give a brief description of the Feistel ciphers that will be investigated in this thesis, they are listed in Table 2.1.

Table 2.1: Specific examples of Feistel structures

Cipher	# branches and f	Type of f
CLEFIA	GFN_4-II	Substitution-Permutation
Camellia	GFN_2	Substitution-Permutation
SMS4	GFN_4-I	Substitution-Permutation
CAST-256	GFN_4-I	Substitution-Permutation
SIMON	GFN_2	AND,Rotation,XOR

CLEFIA is an important block cipher designed by Sony Corporation. It has been accepted as a lightweight encryption algorithm of the ISO/IEC 29192-2 standard [5]. CLEFIA is a 4-line type-II GFN (GFN_4-II) with DSM and single SD-function, here DSM (diffusion switching mechanism) means that it employs two distinct diffusion matrices in each two rounds, and single SD-function means that the round function consists of a single Substitution-Diffusion (SD) layer. We discuss the security of CLEFIA-type GFNs in Section 4.1.4.

CAST-256 is a first-round AES candidate. The design of CAST-256 is a generalized Feistel network with 4 lines (GFN_4). The round function adopts SP structure. The security is discussed in Section 4.2.3.

Camellia was jointly proposed by NTT and Mitsubishi in 2000. It has been selected as one of the CRYPTREC e-government recommended ciphers [55], and included in the NESSIE block cipher portfolio [131]. In 2005, it was adopted as international standard by ISO/IEC 18033-3 [4]. Camellia follows two-line Feistel network. Substitution-Permutation structure is used in the round function. We study it in Section 4.2.3.

SMS4 is the Chinese national standard for Wireless Local Area Networks (WLANs). The structure of SMS4 is GFN_4-I, and the round function is an Substitution-Permutation function. We analyse the security of SMS4 in Section 4.2.3.

SIMON [15], a lightweight design by NSA, follows two-branch Feistel structure. The round function only uses three operations AND, Rotation and XOR. The member of SIMON with block size $n \in \{32, 48, 64, 96, 128\}$ is denoted as SIMONn in this thesis. We discuss our cryptanalytic results for SIMON in Section 4.2.2.

2.2.4 Lai-Massey schemes

Other than Feistel networks or SPNs, one can also follow the third design strategy in which incompatible group operations such as modular addition and modular multiplication are used in the round function. A prominent example is the block cipher IDEA [101] designed by Lai and Massey. Here we call it Lai-Massey schemes. As in Feistel networks, Lai-Massey schemes also divide the input into two equal pieces of $n/2$ bits. For iterated r rounds, two functions (H is invertible, but f is not necessarily) are used in each round:

$$H : \mathbb{F}_2^n \to \mathbb{F}_2^n \ ,$$

$$f : \mathbb{F}_2^{n/2} \to \mathbb{F}_2^{n/2} \ .$$

The function H is called the half-round function. In the ith round, the input state (L_{i-1}, R_{i-1}) is updated as

$$\begin{aligned}(L'_{i-1}, R'_{i-1}) =& H(L_{i-1}, R_{i-1}) \ , \\ T_{i-1} =& f(L'_{i-1} \boxminus R'_{i-1}, k_i) \ , \\ (L_i, R_i) =& (L'_{i-1} \boxplus T_{i-1}, R'_{i-1} \boxplus T_{i-1}) \ ,\end{aligned}$$

where $1 \leq i \leq r$. One round of the Lai-Massey scheme is shown in Figure 2.7.

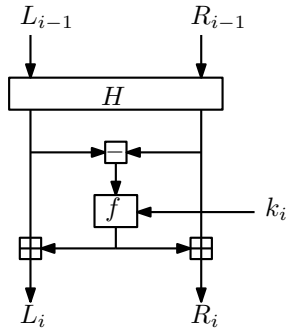


Figure 2.7: One round Lai-Massey scheme

2.3 Modes of operation

So far the encryption functions we have described are always mapping from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$, i.e. permutations operate on an n-bit block size input. However, bit string to be encrypted might have a size larger than the block size n. We need to find a way to extend the application of block ciphers. The solution is to use a block cipher mode of operation. If the given message is a string of size larger than n, then we split it into n-bit blocks. If the length of the last part of the string is smaller than n, a padding scheme may be applied to make the input length exact multiple of n.

The most popular block cipher mode of operation is cipher block chaining (CBC) with the AES block cipher. It was one of the most widely used modes of operations for the TLS protocol, however it is no longer recommended because of the padding oracle attack by Vaudenay [160]. Other prominent modes are electronic code book (ECB) which is insecure and never recommended for use, ciphertext feed back (CFB) mode and counter (CTR) mode which make a block cipher into a stream cipher.

The block cipher modes mentioned here can provide confidentiality, but they do not offer data integrity, as described in Chapter 1. A separate message authentication code such as CBC-MAC [72], or a digital signature can achieve these two purpose by itself. The cryptographic community is designing modes that combine confidentiality and data integrity into a single cryptographic primitive. One example is the ongoing CAESAR competition in Section 1.2.5.

2.4 Conclusions

This chapter introduced the background of block ciphers: the definitions, the design methods and the mode of operations. The most widely used design methods: the generalised Feistel networks and Substitution-Permutation networks of iterated design strategy are explained. The block ciphers that we analyse in this thesis all belong to these two categories, are briefly described in this chapter.

Chapter 3

Cryptanalysis Methods

This chapter provides a brief introduction to the cryptanalytic methods developed for block ciphers. Even if we explain all the cryptanalysis methods focusing on block ciphers, we note that they can be applied to other symmetric-key primitives. This chapter is organized as follows. First the attack models are explained in Section 3.1. Then some known cryptanalysis methods of block ciphers are briefly described, especially the ones we are applying in this thesis. We start with differential cryptanalysis and its extensions: impossible-differential cryptanalysis, the boomerang attack, the rectangle attack, higher-order differential attack and integral attack, in Sections 3.2. Then we describe linear cryptanalysis and its extension zero-correlation linear cryptanalysis in Section 3.3. Finally we explain algebraic cryptanalysis and interpolation attack in Section 3.4.

3.1 Attack models

3.1.1 Attack data types

In the cryptanalysis of block ciphers, the objective of the attacker is to perform operations that should only be possible with knowledge of the secret key, for example to determine the encryptions/decryptions of messages previously unknown to the attacker. Depending on the increasing power of the information available to the adversary, the attack models can be defined as follows:

Ciphertext-Only (CO) attack: The attacker can only get access to

the ciphertexts, and some statistical information on the plaintext, such as the fact that the plaintext is written in English. Encryption vulnerable to this type of attack is considered to be completely insecure.

Known-Plaintext (KP) attack: The attacker has a quantity of plaintexts and the corresponding ciphertexts. This type of attack is a passive attack since the attacker cannot influence the content of the plaintext.

Chosen-Plaintext (CP) attack: The attacker can choose the plaintext and get the corresponding ciphertext.

An *adaptive Chosen-Plaintext attack* is a Chosen-Plaintext attack wherein the choice of the next plaintext in a series may depend on the ciphertexts received from the previous requests.

Chosen-Ciphertext (CC) attack: The attacker can choose the ciphertext and get the corresponding plaintext.

An *adaptive Chosen-Ciphertext attack* is a Chosen-Ciphertext attack wherein the choice of the next ciphertext in a series may depend on the plaintexts received from the previous requests.

Chosen-Plaintext and Chosen-Ciphertext attack: The attacker is able to not just receive the encryption of chosen plaintexts, but can also obtain the decryption of chosen ciphertexts.

Similarly, in an *adaptive Chosen-Plaintext and Chosen-Ciphertext attack*, the attacker chooses plaintext and ciphertext based on the outcome of earlier requests.

The attack models described here are not limited to block ciphers, but can be applied to other cryptographic primitives and protocols.

3.1.2 Attack key settings

Besides the attack data types described above, models of the attacker's knowledge about the secret key is also considered for crytanalysis. When explaining an attack, we use one of the models described above under one of the following key settings:

Secret-key/single-key model: The adversary does not know any information about the secret key K.

Related-key model: In the related-key model, the adversary can decrypt/encrypt not only under the secret key K, but also under the keys $f_1(K), f_2(K) \dots f_m(K)$, which are called related-keys. The relations f_i are chosen by the adversary in advance. The first related-key attacks consider simple mappings, for example, rotations [22] and bit flips [95]. The later attack on AES [32] exploits the difference not between the secret keys but between the subkeys. Note that no cipher can reach full secure against related-key attacks. The extra control might make the attack harder to mount in practice. However, great efforts are still being made to obtain a related-key attack since it is the primary step to evaluate the security of a cipher, based one which, at some time, an improved attack might be achieved.

Known-key model: In the known-key model, the secret key is revealed to the adversary. Block ciphers are indeed used in a setting where the secret key is known to the public. Hash functions which build on block ciphers, including the Davies-Meyer (DM), Matyas-Meyer-Oseas (MMO) and Miyaguchi-Preneel (MP) modes are such examples [13, 14, 75]. In the known-key model, the adversary aims to show non-ideal property of the design, which is not provided by the randomness of particular secret keys.

Chosen-key model: Similar to the known-key model, the adversary has knowledge of the secret key, and has the same goal to find the non-ideal property of the cipher. Unlike the known-key model, the adversary is not only given the secret key, but can also freely choose the key herself to serve her goal.

3.1.3 Attack goals

The most severe attack for a symmetric-key primitive is that the adversary can recover the secret key; in this case the symmetric-key primitive is totally broken. In some cases, the adversary might be less ambitious, and she/he might just gain more information than expected. A hierarchy of the adversary's goals is follows [98]:

Key recovery: The adversary is able to determine the secret key of the symmetric primitive, therefore she can decrypt any ciphertext encrypted with this secret key.

Global deduction: The attacker is able to determine a function which is functionally equivalent to the encryption/decryption of the algorithm, without knowing the secret key.

Local deduction: The adversary can generate the message (or ciphertext) corresponding to a previously unseen ciphertext (or message).

Distinguisher: Intuitively, a well-designed symmetric-key primitive behaves like a random permutation. If the primitive is distinguishable, it means that the adversary can efficiently distinguish between two black boxes; one contains the symmetric-key primitive with a randomly chosen encryption key while the other contains a randomly chosen permutation.

We need to point out that based on a distinguisher of the cipher, the adversary might mount an advanced attack. This can be done by first collecting sufficiently many pairs of plaintexts and ciphertexts as required by the distinguisher, then guessing a part of the secret key to do partial decryption and applying the distinguisher to a set of the pairs (encrypted plaintext, and the corresponding partially decrypted ciphertext). If the distinguisher holds, the guess is correct, otherwise it is wrong. Thus, when evaluating the resistance of the new design against the existing attack methodologies, this provides a strong basis to the designers to make security claims, as long as constructing such distinguishers is impossible. If it is not possible to construct distinguishers, obviously it is even harder to achieve the rest of the attack goals.

3.1.4 Attack complexity metrics

To estimate the complexity of a cryptanalytic attack, one must take into account at least three metrics: the time it takes, the amount of data it needs (when the type of data is fixed as described in subsection 3.1.1), and the storage it requires.

Data Complexity: The amount of data needed as input to an attack. The data can be measured in terms of the block size. Also the data complexity should be referred to the attack data types: CO, KP, CP, CC, etc. When the data complexity covers all pairs of plaintext and ciphertext, we call the attack the full code book attack.

Time Complexity: The time needed to perform an attack. The time complexity can be measured by several units, for example, CPU instructions, or number of encryptions. When analysing time complexity, we consider both the *offline* and *online* phase of an attack. In the offline phase, the complexity is the time required for the attack before obtaining any plaintext/ciphertext pair (which is also called *precomputation*), while in the online phase, the complexity is the time to obtain the data required by the attacker.

Memory Complexity: The amount of memory needed for the attack. It can be measured by various units, for example, the block size (bytes), the hash table size, etc. Sometimes the memory complexity is so large that it makes the attack completely impractical.

The complexity of an attack is often evaluated as the maximum of the three complexities above; however, in most cases the amount of data the adversary can obtain with the same secret key is limited (confined by the designers); on the other hand for most attackers the available storage is small (< 1 PB).

Before we start to describe the crytanalysis methods of block ciphers, we note that throughout this chapter, E (defined in Section 2.1) can be treated as a cascade of two sub-ciphers $E = E_1 \circ E_0$ if necessary. When appending some rounds before and/or after E in order to mount an r'-round attack, we denote the new targeted cipher as E'.

3.2 Differential cryptanalysis and extensions

3.2.1 Differential cryptanalysis

Differential cryptanalysis was introduced by Biham and Shamir [28] in 1991. It is one of the most powerful methods for the cryptanalysis of symmetric-key primitives; it resulted in the first attack on full DES [29], and the first related-key attack on full AES-192 and AES-256 [32, 33]. The main idea of differential cryptanalysis is to study the difference propagations through the rounds of the cipher E. For common cases the difference is an XOR difference, sometimes a modular subtraction [104] is used. Differential cryptanalysis is a Chosen-Plaintext attack that is performed in two phases: *data collection* and *key recovery* phase.

Data collection Let P and P' be a pair of plaintexts of E with input difference ΔP, and let C and C' be the corresponding ciphertexts satisfying the output difference ΔC. The pair $(\Delta P, \Delta C)$ is called an r-round *differential* (denoted as $(\Delta P \to \Delta C)$ as well) where the input difference ΔP propagates to the output difference ΔC. The fraction of pairs (P, P') satisfying $(\Delta P, \Delta C)$ after r rounds is called the *differential probability* (DP) of the r-round differential, denoted as $\mathrm{DP} = \Pr(\Delta P, \Delta C)$, i.e.

$$\mathrm{DP}(\Delta P, \Delta C) = \frac{1}{2^n} \#\{P \in \mathbb{F}_2^n | E(P \oplus \Delta P) = E(P) \oplus \Delta C\} \ .$$

If E is keyed by a key k, we write $\mathrm{DP}(\Delta P, \Delta C; k)$, and the *expected differential probability* $\mathrm{EDP}(\Delta P, \Delta C)$ is defined as

$$\mathrm{EDP}(\Delta P, \Delta C) = E_K[\mathrm{DP}(\Delta P, \Delta C; K)] \ ,$$

where K is uniformly distributed over the key space $\mathbb{F}_2^k$.

A *differential characteristic, trail*, or *path* is a vector $(\alpha_0, \alpha_1, \ldots, \alpha_r)$ with α_i specifying the difference after the ith round. A differential characteristic $T = (\alpha_0, \alpha_1, \ldots, \alpha_r)$ is said to be part of differential $(\Delta P, \Delta C)$, denoted as $T \in (\Delta P, \Delta C)$, if $\alpha_0 = \Delta P$ and $\alpha_r = \Delta C$. Therefore, a differential is the collection of all the differential characteristics with the same input and output difference. $\mathrm{DP}(\Delta P, \Delta C)$ is the sum over the probabilities of all the differential characteristics belonging to $(\Delta P, \Delta C)$:

$$\mathrm{DP}(\Delta P, \Delta C) = \sum_{T \in (\Delta P, \Delta C)} \mathrm{DP}(T) = \sum_{\alpha_1, \ldots, \alpha_r} \prod_{i=1}^{r} \mathrm{DP}(\alpha_{i-1}, \alpha_i; k_i) \ ,$$

where k_i is the subkey for the ith round of E. Similarly,

$$\mathrm{EDP}(\Delta P, \Delta C) = \sum_{T \in (\Delta P, \Delta C)} \mathrm{EDP}(T) = \sum_{\alpha_1, \ldots, \alpha_r} \prod_{i=1}^{r} \mathrm{EDP}(\alpha_{i-1}, \alpha_i) \ ,$$

The *maximum differential probability* (MDP) is the maximum value of the differential probabilities over all pairs of non-zero input difference:

$$\mathrm{MDP} = \max_{\Delta P \neq 0, \Delta C} \mathrm{DP}(\Delta P, \Delta C) \ .$$

Similarly, the *maximum expected differential probability* (MEDP) is defined as

$$\mathrm{MEDP} = \max_{\Delta P \neq 0, \Delta C} \mathrm{EDP}(\Delta P, \Delta C) \ .$$

The goal of this phase is to find an r-round differential with high DP. If DP is relatively high, it can be used to recover the secret key.

Key recovery Based on an r-round differential with high DP, one might attack at least $r+1$ rounds depending on the structure of the cipher. For instance one extra round at the end is considered: after guessing some subkey bits of the last round, the adversary partially decrypts one round to check whether the expected differential is observed or not after the rth round. If yes, a counter for the guessed key is increased by one. The guessed key with the highest occurrence is selected as a correct key.

Many variants have developed from differential cryptanalysis; the applications of these methods are not only limited to analysing block ciphers, but are crucial in the analysis of almost all symmetric-key components, for example hash functions. We will briefly describe the ones related to this thesis in sequence.

3.2.2 Impossible-differential cryptanalysis

Impossible-differential cryptanalysis was independently proposed by Borst et al. to attack IDEA [46] and by Biham et al. to attack Skipjack [24]. The technique was applied later in the analysis of other block ciphers including AES [130], Camellia [48, 109] and CLEFIA [159].

In differential cryptanalysis the attacker tries to find a differential that occurs with high probability, whereas in impossible-differential cryptanalysis the attacker aims to construct a differential that does not occur (i.e. with probability 0). The approach to construct such a differential is based on the idea of miss-in-the-middle. For cipher E, assume there exist a differential $(\alpha \rightarrow \beta)$ of E_0 and $(\gamma \rightarrow \delta)$ of E_1^{-1}, both with probability one. However, $\beta \neq \delta$ contradicts the differential $(\alpha \rightarrow \gamma)$, thus an r-round impossible-differential of E, denoted as $(\alpha \not\rightarrow \gamma)$, is obtained.

During the key recovery phase, by guessing some subkey bits of the appended rounds in E', the attacker performs partial encryption and/or decryption to collect a sufficient number of pairs with specific input and output differences of E. If there exists a pair with input difference α and output difference γ of the impossible-differential under some subkey bits, the attacker is assured that the key guess must be wrong. In this way, many wrong keys are discarded. Depending on the cipher and the key schedule, the number of remaining candidates of the right key might be one, while in some case this might be a large number; subsequently the attacker performs an exhaustive search of the remaining key(s). The construction of an impossible-differential and the key recovery procedure are depicted in Figure 3.1. We apply impossible-differential cryptanalysis method to analyse block ciphers Rijndael-224, Rijndael-256 and SIMON (refer to Section 2.2.2 and 2.2.3) in Section 4.2.

3.2.3 Boomerang attack

The boomerang attack proposed by Wagner [161], aims to reduce the complexity of differential cryptanalysis. The main idea of the attacks is to use two short differentials with high probabilities instead of one differential of more rounds with low probability. The motivation for such an attack is that for many ciphers

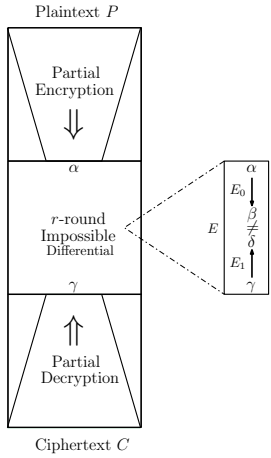


Figure 3.1: Impossible-differential cryptanalysis

it is easier to find short differentials with high probabilities than finding a longer one with a relatively high probability. This attack is a Chosen-Plaintext and Chosen-Ciphertext attack.

The attack is based on a quartet of encryptions and decryptions. A right *quartet* of E is defined as the plaintext (P_1, P_2, P_3, P_4) and ciphertext (C_1, C_2, C_3, C_4) satisfying the following conditions:

$$P_1 \oplus P_2 = P_3 \oplus P_4 = \alpha\,, \quad E_0(P_1) \oplus E_0(P_2) = E_0(P_3) \oplus E_0(P_4) = \beta\,,$$

$$C_1 \oplus C_3 = C_2 \oplus C_4 = \gamma\,, \quad E_0(P_1) \oplus E_0(P_3) = E_0(P_2) \oplus E_0(P_4) = \delta\,,$$

where $E = E_1 \circ E_0$ and $E_0(P_i) = E_1^{-1}(C_i)$ for $1 \leq i \leq 4$. A right quartet is depicted in Figure 3.2.

In order to build a right quartet for the block cipher E such that there exists a differential $(\alpha \to \beta)$ with probability p for E_0, and a differential $(\gamma \to \delta)$ with probability q for E_1^{-1}, the attacker performs the following steps:

1. Ask for the encryption of a plaintext pair (P_1, P_2) where $P_1 \oplus P_2 = \alpha$, and denote the corresponding ciphertexts by (C_1, C_2).
2. Compute the ciphertext pair (C_3, C_4) that satisfies $C_3 = C_1 \oplus \gamma$ and $C_4 = C_2 \oplus \gamma$. Ask for the decryption of (C_3, C_4) and denote the corresponding plaintexts by (P_3, P_4).
3. Check whether $P_3 \oplus P_4 = \alpha$.

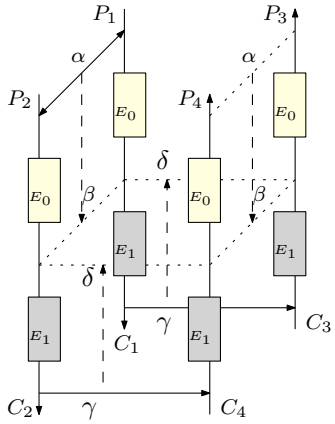


Figure 3.2: A right quartet of the boomerang distinguisher

For a random permutation, the condition $P_3 \oplus P_4 = \alpha$ is satisfied with probability 2^{-n}. For E, the probability that the pair (P_1, P_2) is a right pair of the differential $(\alpha \to \beta)$ is p, the probability that both pairs (C_1, C_3) and (C_2, C_4) are right pairs of the differential $(\gamma \to \delta)$ is q^2. If all these pairs are right pairs, and they satisfy $E_1^{-1}(C_3) \oplus E_1^{-1}(C_4) = \beta = E_0(P_3) \oplus E_0(P_4)$, then $P_3 \oplus P_4 = \alpha$ holds with probability p. Therefore, the probability that the quartet (P_1, P_2, P_3, P_4) of plaintexts and (C_1, C_2, C_3, C_4) of ciphertexts make up a boomerang distinguisher is $(pq)^2$. If $pq > 2^{-n/2}$ holds, the boomerang distinguisher and the key recovery attacks can work.

Moreover, the attack can be mounted for all possible β's and δ's simultaneously (as long as $\beta \neq \delta$), therefore a right quartet for E can be constructed with probability $(\hat{p}\hat{q})^2$ where

$$\hat{p} = \sqrt{\sum_{\beta} \Pr^2[\alpha \to \beta]} \quad \textit{and} \quad \hat{q} = \sqrt{\sum_{\delta} \Pr^2[\gamma \to \delta]} \, .$$

In addition, it is also possible to combine the boomerang attack with related-key attacks; examples can be found in [27, 32, 33].

3.2.4 Rectangle attack

As the boomerang attack is a Chosen-Plaintext and Chosen-Ciphertext attack, many techniques that developed for using boomerang distinguishers in key

recovery attacks cannot be applied. This leads to the development of a Chosen-Plaintext variant of the boomerang attack: the amplified boomerang attack [94] and the rectangle attack [26]. Because the rectangle attack achieves a larger improvement than the amplified boomerang attack, we only describe the rectangle attack in this thesis, and use the same notation as in Section 3.2.3. The rectangle attack works in a Chosen-Plaintext setting.

A right rectangle quartet of E is defined to be a quartet of plaintexts (P_1, P_2, P_3, P_4) and the corresponding ciphertexts (C_1, C_2, C_3, C_4) such that $P_1 \oplus P_2 = P_3 \oplus P_4 = \alpha$ and $C_1 \oplus C_3 = C_2 \oplus P_4 = \gamma$, where α is the input difference of E_0 and γ is the output difference of E_1^{-1}. These conditions are referred to as the rectangle conditions. For clarity, we denote the partial encryption of P_i by X_i, i.e., $X_i = E_0(P_i) = E_1^{-1}(C_i)$.

The first improvement the rectangle distinguisher makes is in E_1. Instead of using only one value of δ, one can use any δ' that satisfies $\gamma \to \delta'$ for E_1^{-1}, as long as both pairs (X_1, X_3) and (X_2, X_4) have the same difference δ'. In this case (where $X_1 \oplus X_3 = X_2 \oplus X_4 = \delta'$ and $X_1 \oplus X_2 = \beta$), the probability that the rectangle conditions are satisfied is

$$2^{-n} p^2 \sum_{\gamma \xrightarrow{E_1^{-1}} \delta'} \Pr^2[\gamma \xrightarrow{E_1^{-1}} \delta'] = p^2 \hat{q}^2 / 2^n \ .$$

The second improvement similar to the previous one but in E_0, is to use all possible β' values. As long as $X_1 \oplus X_2 = X_3 \oplus X_4$ and $X_1 \oplus X_3 = \delta'$, the probability that a quartet becomes a right quartet is $\Pr^2[\delta' \to \gamma]$. In all, the probability that a given quartet is a right quartet is

$$2^{-n} \sum_{\alpha \xrightarrow{E_0} \beta'} \Pr^2[\alpha \xrightarrow{E_0} \beta'] \cdot \sum_{\gamma \xrightarrow{E_1^{-1}} \delta'} \Pr^2[\gamma \xrightarrow{E_1^{-1}} \delta'] = (\hat{p}\hat{q})^2 / 2^n \ .$$

Based on the above improvements, we conclude that given N plaintext pairs we expect to have $N^2(\hat{p}\hat{q})^2/2^n$ right rectangle quartets. Similarly to the boomerang attack, rectangle attack can also be combined with related-key attacks [96, 111].

We discuss the related-key rectangle attack on Rijndael-160/160 and Rijndael-192/192 (see Section 2.2.2 for the description) in Section 4.1.6.

3.2.5 Higher-order differential attack

The concept of higher-order derivatives has been first introduced in cryptography by Lai [102] and applied to differential cryptanalysis by Knudsen [97]. Whereas

standard differential cryptanalysis studies the propagation of the differences between one pair of plaintexts, higher-order differential cryptanalysis was generalised to exploit the propagation of differences between pairs in a larger set having 2^i pairs of element (the $(i+1)$th-order differential).

Definition 2. *([102]) Let $(S,+)$ and $(T,+)$ be Abelian groups. For a function $f: S \to T$, the derivative at a point $a \in S$ is defined as*

$$\Delta_a f(x) = f(x+a) - f(x) \ .$$

The ith derivative of f at the point $(a_1, a_2, \cdots, a_i)$ is then defined as

$$\Delta^{(i)}_{a_1,\ldots,a_i} f(x) = \Delta_{a_i}(\Delta_{a_1,\ldots,a_{i-1}} f(x)) \ .$$

From the definition, the standard differential corresponds to the first order derivatives. It is natural to extend the notation of first-order differential to higher-order differential as follows:

Definition 3. *([97]) A one-round differential of order i for a function f: $S \to T$ is an $(i+1)$-tuple $(\alpha_1, \alpha_2, \ldots, \alpha_i, \beta)$ such that $\Delta^{(i)}_{(\alpha_1,\ldots,\alpha_i)} f(x) = \beta$.*

Proposition 1. *([102]) Let $L[a_1, a_2, \ldots, a_i]$ be the list of all 2^i possible linear combinations of $a_1, a_2, \ldots, a_i$, then*

$$\Delta^{(i)}_{a_1,\ldots,a_i} f(x) = \sum_{\Delta P \in L[a_1,\ldots,a_i]} f(P + \Delta P) \ .$$

If $a_1, \ldots, a_i$ are linearly dependent, then

$$\Delta^{(i)}_{a_1,\ldots,a_i} f(x) = 0 \ .$$

For the block cipher E, assume the algebraic degree (see Appendix A) of E_0 is $d-1$, and E_0 has $r-1$ rounds (from our definition, E_1 has one round). Then

$$\Delta^{(d)}_{a_1,\cdots,a_d} E_0(P, k_1, \ldots, k_{r-1}) = 0 \ , \tag{3.1}$$

where $a_1, \cdots, a_d \in \mathbb{F}_2^{d-1}$. Let E_1^{-1} be the inverse of sub-cipher E_1, then

$$E_1^{-1}(C, k_r) = E_1^{-1}(E(P), k_r) = E_0(P, k_1, \ldots, k_{r-1}) \ . \tag{3.2}$$

From Equation (3.1) and (3.2), one can get the attack equation

$$\Delta^{(d)}_{a_1,\cdots,a_d} E_1^{-1}(C, k_r) = 0 \ . \tag{3.3}$$

By checking the attack equation (3.3), the subkeys involved in the final round E_1 can be obtained.

3.2.6 Integral attack

The integral attack, also known as the square attack [99], was first proposed by Knudsen to analyse SQUARE [57] block cipher. There are several variants of the integral attack with different names: multiset attack [35], saturation attack [112], and collision attack [78]. The integral attack can be seen as the dual of a differential attack; it can be very effective to attack ciphers that are secure against differential attack. For instance, the integral attack is the best known attack on AES.

Let Λ be a collection of state vectors $X = (x_0, \ldots, x_{n-1})$, where $x_i \in \mathbb{F}_2$ is the ith word of X. The following notations are defined:

A: if all ith words x_i in Λ are distinct, x_i is called active.

B: if the sum of all ith words x_i in Λ can be predicted, x_i is called balanced.

C: if the values of all ith words x_i in Λ are equal, x_i is called passive/constant.

*: if the sum of all ith words x_i in Λ cannot be predicted.

The integral attack first constructs an integral distinguisher by choosing plaintexts with some active (A) words, and the rest as constant (C). Then ask for the encryption of the plaintexts and check whether the summation (usually XOR) at some word is 0, if yes, we call this word balanced (B), and an r-round integral distinguisher is found. During the key recovery phase, firstly several final rounds are appended to the distinguisher, next some subkeys involved are guessed in order to do partial decryptions up to the output of the integral distinguisher. If the guess is correct, the XOR sum of the states is always 0, thus the key space can be reduced.

The integral attack is applied to analyse Feistel structures Camellia, CAST-256, SMS4 and SIMON (see Section 2.2.3) in Section 4.2.

3.2.7 Local collision

A local collision is a differential that starts and ends with a zero difference in the internal state, but is non-zero in the intermediate state. The idea of local collisions has been first introduced by Joux and Chabaud [52] to attack hash functions. It aims to inject a difference (called *disturbance*, in red) into an intermediate step and then to *correct* the resulting differences with injections in the next steps (called *correction*, in grey) to obtain a collision. The goal is to reduce the complexity of the attack by having as few disturbances as

possible. This idea has been later applied by Biryukov and Khovratovich [32] to attack AES-192 and AES-256 under the related-key model. A local collision of AES-256 is shown in Figure 3.3.

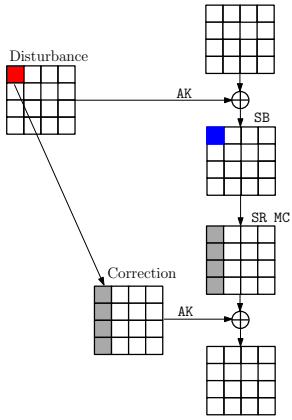


Figure 3.3: A local collision in AES-256

We use this idea to construct related-key rectangle distinguishers for Rijndael-160/160 and Rijndael-192/192 (refer to Section 2.2.2) in Section 4.1.6.

3.3 Linear cryptanalysis and extensions

3.3.1 Linear cryptanalysis

Linear cryptanalysis, along with differential cryptanalysis, is one of the most powerful cryptanalysis techniques for symmetric-key primitives. The idea was used to attack the block cipher FEAL [123] by Tardy-Corfdir and Gilbert [158] in 1991, and Matsui and Yamagishi [116] in 1992; later it has been successfully applied to DES [114].

The linear attack is a Known-Plaintext attack in which the adversary tries to find a linear relation between plaintext and ciphertext that holds with a sufficiently large linear probability. Let Γ_P and Γ_C denote the masks of P and C respectively. Assume that $\Gamma_P \cdot P \oplus \Gamma_C \cdot C = 0$ holds with *linear probability*

$\mathrm{LP}(\Gamma_P, \Gamma_C)$ as

$$\mathrm{LP}(\Gamma_P, \Gamma_C) = \left(\frac{1}{2^{n-1}} \#\{P \in \mathbb{F}_2^n | \Gamma_P \cdot P \oplus \Gamma_C \cdot C = 0\} - 1 \right)^2 ,$$

we call $(\Gamma_P \to \Gamma_C)$ (also denoted as (Γ_P, Γ_C)) a *linear approximation/hull* of E with linear probability $\mathrm{LP}(\Gamma_P, \Gamma_C)$. If E is keyed by a key k, we write $\mathrm{LP}(\Gamma_P, \Gamma_C; k)$, and the *expected linear probability* $\mathrm{ELP}(\Gamma_P, \Gamma_C)$ is defined as

$$\mathrm{ELP}(\Gamma_P, \Gamma_C) = E_K[\mathrm{LP}(\Gamma_P, \Gamma_C; K)] ,$$

where K is uniformly distributed over the key space $\mathbb{F}_2^k$.

We define the *linear correlation* c of $(\Gamma_P \to \Gamma_C)$ as

$$c = \frac{1}{2^n} (\#\{P \in \mathbb{F}_2^n | \Gamma_P \cdot P \oplus \Gamma_C \cdot C = 0\} - \#\{P \in \mathbb{F}_2^n | \Gamma_P \cdot P \oplus \Gamma_C \cdot C = 1\}) .$$

As can be seen, $\mathrm{LP} = c^2$.

Lemma 1 (Piling-up Lemma [114])**.** *Let X_i $(1 \le i \le n)$ be independent binary random variables whose values are 0 with correlation c_i. Then the correlation of $X_1 \oplus X_2 \oplus \cdots \oplus X_n = 0$ is*

$$c = \prod_{i=1}^{n} c_i .$$

Let $X_0, X_1, \ldots, X_r$ denote the intermediate values of E where $X_i = E(X_{i-1})$, $P = X_0$ and $C = X_r$. Let $\Gamma_{X_{i-1}}$ and Γ_{X_i} denote the input and output masks respectively for the ith round $(1 \le i \le r)$. A pair of masks $(\Gamma_{X_{i-1}}, \Gamma_{X_i})$ is called a one-round *linear characteristic* for the ith round, and the $(r+1)$-tuple $\Omega = (\Gamma_{X_0}, \Gamma_{X_1}, \ldots, \Gamma_{X_r})$ is called an r-round linear characteristic. A linear characteristic Ω is called to be part of linear approximation (Γ_P, Γ_C), denoted as $\Omega \in (\Gamma_P, \Gamma_C)$, if $\Gamma_{X_0} = \Gamma_P$ and $\Gamma_{X_r} = \Gamma_C$. Therefore, a linear approximation is the collection of all the linear characteristics with the same input and output mask. Assume that these one-round linear characteristics are statistically independent, then $\mathrm{LP}(\Gamma_P, \Gamma_C)$ and $\mathrm{ELP}(\Gamma_P, \Gamma_C)$ can be calculated by Piling-up lemma as:

$$\mathrm{LP}(\Gamma_P, \Gamma_C) = \sum_{\Omega \in (\Gamma_P, \Gamma_C)} \mathrm{LP}(\Omega) = \sum_{\Gamma_{X_1}, \ldots, \Gamma_{X_r}} \prod_{i=1}^{r} \mathrm{LP}(\Gamma_{X_{i-1}}, \Gamma_{X_i}; k_i) ,$$

where k_i is the subkeys being used for the ith round of E, and

$$\mathrm{ELP}(\Gamma_P, \Gamma_C) = \sum_{\Omega \in (\Gamma_P, \Gamma_C)} \mathrm{ELP}(\Omega) = \sum_{\Gamma_{X_1}, \ldots, \Gamma_{X_r}} \prod_{i=1}^{r} \mathrm{ELP}(\Gamma_{X_{i-1}}, \Gamma_{X_i}) .$$

The *maximum linear probability* (MLP) is the maximum value of the linear probabilities over all pairs of non-zero input and output mask:

$$\mathrm{MLP} = \max_{\Gamma_P, \Gamma_C \neq 0} \mathrm{LP}(\Gamma_P, \Gamma_C) \ .$$

Similarly, the *maximum expected linear probability* (MELP) is the maximum value of the expected linear probabilities over all non-zero input and output mask:

$$\mathrm{MELP} = \max_{\Gamma_P, \Gamma_C \neq 0} \mathrm{ELP}(\Gamma_P, \Gamma_C) \ .$$

After obtaining a linear approximation with sufficiently large LP, an adversary can mount a key recovery attack on E' by guessing some subkey bits used in the appended rounds. Matsui [114] showed that the number of plaintext-ciphertext pairs required in the key recovery attack can be estimated as $4c_N \times c^{-2}$, where c_N is related to the number of guessed subkey bits. The success rate can be calculated by the formula proposed by Selçuk [146].

Several extensions of linear cryptanalysis have been proposed: one approach is to combine differential and linear cryptanalysis [25, 37, 105, 110]; multiple linear cryptanalysis combines more than one linear approximations [31, 83, 91, 128].

3.3.2 Zero-correlation linear cryptanalysis

The zero-correlation attack can be treated as the counterpart of impossible-differential attack in linear cryptanalysis. While impossible-differential cryptanalysis has been known for about 20 years, zero-correlation attack was only proposed by Bogdanov and Rijmen [43] in 2011; it has been shown to be a promising attack technique for block ciphers [44].

The first step for the zero-correlation attack is to construct a linear distinguisher with correlation zero by adopting the miss-in-the-middle techniques as that is used in impossible-differential cryptanalysis. Assume for the block cipher E, we obtain a linear approximation $(\Gamma_\alpha \to \Gamma_\beta)$ with correlation zero. Based on this r-round zero correlation linear approximation $(\Gamma_\alpha \to \Gamma_\beta)$, one can distinguish E from an ideal n-bit block cipher since for the latter, the probability that $(\Gamma_\alpha \to \Gamma_\gamma)$ has correlation zero is about $\frac{1}{\sqrt{2\pi}} 2^{\frac{4-n}{2}} \approx 0$ (if $n \geq 32$), while for the former, the probability that $(\Gamma_\alpha \to \Gamma_\beta)$ has correlation zero is 1. Moreover, during the key recovery phase, based on the above zero correlation linear approximation, an adversary can mount a key recovery attack on E' by means of guessing part of subkeys adopted in the appended rounds of E.

Zero-correlation linear cryptanalysis is used to analyse SIMON in Section 4.2. The links between impossible-differential, zero-correlation and integral attack are discussed there as well.

3.4 Algebraic cryptanalysis

Compared to differential and linear cryptanalysis, algebraic cryptanalysis [54] is a relatively new technique for the analysis of cryptographic primitives; it was applied more successfully to analyse stream ciphers [54, 69]. The main idea of algebraic cryptanalysis is to first express a cipher as a system of algebraic nonlinear equations in terms of the secret key, and then to solve the equations in order to recover the secret key.

A standard method to solve a system of linear equations is Gaussian elimination. However, the equations describing the cryptographic primitives are highly nonlinear. An efficient approach to this problem is called *linearization*, where all monomials of degree larger than one are replaced by a new independent variable of degree one. In this way the system of nonlinear equations is transformed into a system of linear equations which can then be solved by Gaussian elimination.

Another direction to solve the nonlinear system is to use either a SAT solver or Gröbner bases. For a SAT solver, the system of equations is represented as a *Boolean Satisfiability* problem; a number of heuristic off-the-shelf solvers [3], for instance Cryptominisat [124], are available to solve the system. The Gröbner bases [8] technique enables to significantly reduce the degree of the nonlinear polynomials by multiplying them with well-chosen multivariate polynomials. Then, it may be able to recover the key by solving the system of equations. However, in many cases the time and memory of this algorithm is prohibitively large.

3.4.1 Interpolation attack

The interpolation attack was introduced by Jakobsen and Knudsen [87, 88] as an algebraic attack on block ciphers that are built from low degree algebraic functions. Given an unknown polynomial $y = f(x)$, if the degree of $f(x)$ does not exceed $n - 1$, then its coefficients can efficiently be recovered by taking n distinct samples (x_i, y_i) with $y_i = f(x_i)$. By using the Lagrange interpolation formula, the polynomial can be reconstructed as

$$f(x) = \sum_i y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} \ .$$

When applying the interpolation attack to a block cipher, we represent the ciphertext (or the intermediate target) as a polynomial of the plaintexts and the key as

$$f(P,K) = f_K(P_1,\ldots,P_n) = \sum_{u\in\mathbb{F}_{2^n}} \alpha_u M_u \ ,$$

where $\alpha_u \in \{0,1\}$ and $M_u = \prod_{i=1}^{n} P_i^{u_i}$. The coefficient α_u depends on the key and are unknown. The goal of the interpolation attack is to interpolate the unknown coefficients of $f(P,K)$ and then eventually recover the secret key.

The interpolation attack is successfully applied to analyse LowMC in Section 4.3.

3.5 Conclusions

In this chapter, we first introduced attack models of block ciphers, by which we can describe an attack in aspects of the key setting, the attack goal, the data type and the complexity. We mainly focused on the most common cryptanalysis methods that are necessary to understand our results in the next chapters.

Chapter 4

Contributions of this thesis

This chapter presents a survey of our contributions on the cryptanalysis and design of symmetric-key primitives. In Section 4.1, mixed integer linear programming (MILP) and its applications to symmetric-key primitives are introduced. In Section 4.2, we first present our impossible-differential cryptanalysis results of Rijndael-224 and Rijndael-256, and then we describe the integral and zero-correlation linear attacks on SIMON. Subsequently, the links among impossible-differential, zero-correlation and integral cryptanalysis are discussed. Finally Section 4.3 introduces our attack on the new block cipher LowMC.

4.1 MILP and its applications to symmetric-key primitives

In this thesis, we apply the MILP technique to evaluate the security of symmetric-key primitives against differential/linear cryptanalysis in the single-key and/or related-key models. Applications include the analysis of stream cipher Enocoro-128v2, generalized Feistel networks GFN_4-II, Rijndael-160 and Rijndael-192 in the related-key model, and the design of the underlying permutation for the second round CAESAR candidate PRIMATEs.

4.1.1 Backgrounds

Linear programming (LP) is the study of optimizing (minimizing or maximizing) a linear objective function $f(x_1, x_2, \ldots, x_n)$, subject to linear inequalities involving decision variables x_i, $1 \leq i \leq n$. For many such optimization problems, it is necessary to restrict certain decision variables to integer values, i.e. for some values of i, we require $x_i \in \mathbb{Z}$. Methods of formulating and solving such programs are called mixed-integer linear programming (MILP). If all decision variables x_i must be integers, the term (pure) integer linear programming (ILP) is to be used. MILP techniques have found many applications in the fields of economy and business, but their application in cryptography has been limited. We proposed our MILP-based technique [125] in 2011. Similar techniques were applied to search the differentials and linear hulls of a large number of block ciphers with SPN and GFN structures, e.g. PRESENT, SIMON, LBlock and DES(L) [156], and to the design of the permutation of the AE scheme such as Prøst [93].

Resistance against linear and differential cryptanalysis is a standard design criterion for new ciphers. For the AES [60], provable security against linear and differential cryptanalysis follows from the wide trail design strategy. We apply a similar proof strategy: After proving a lower bound on the number of active S-boxes for both differential and linear cryptanalysis, we use the maximum differential/linear probability of the S-boxes to derive an upper bound for the probability of the best characteristic. As is commonly done, the probability of the differential/linear hull is estimated by the probability of the best characteristic. Therefore the main task goes to calculate the minimum number of active S-boxes.

Several works focus on calculating the minimum number of active S-boxes for both SPNs [59] and GFNs [40, 41, 47, 92, 151, 172]. Unfortunately, it seems that a large amount of time and programming effort is required to apply those techniques, because they require solving several ILP problems. We show how this can be avoided by introducing extra dummy variables into the MILP to generate one single ILP problem, achieving the fully automatic search for differential and linear characteristics.

4.1.2 Applications to stream cipher Enocoro-128v2

Enocoro-128v2 [169] is a lightweight stream cipher inspired by the Panama construction [56]. It has been adopted as ISO standard [6] in 2012. There are 96 initialization rounds in Enocoro-128v2. The internal state of Enocoro-128v2 is composed of a buffer consisting of 32 bytes and a state consisting of two

bytes; it is initialized with a 128-bit key and a 64-bit IV. The update function deploys an 8-bit S-box and a linear transformation with branch number 3. We apply our technique to Enocoro-128v2 to obtain bounds against differential and linear cryptanalysis [125]. All MILP problems are solved using the CPLEX solver [1]. We prove that 38 rounds are sufficient for security against differential cryptanalysis, and 61 rounds against linear cryptanalysis. These security bounds are obtained after about 53 and 229 seconds respectively on a 24-core Intel Xeon X5670 Processor with 16 GB of RAM.

4.1.3 Applications to AES

We also calculate the minimum number of active S-boxes for up to 14 rounds of AES, which takes at most 0.40 seconds for each optimization program [125]. Our experiments are performed on a 24-core Intel Xeon X5670 Processor, with 16 GB of RAM. Our technique can be applied to evaluate all the AES-like symmetric-key primitives against differential and linear cryptanalysis directly. In fact, we apply this technique in deriving the differential and linear bounds of the underlying permutation of PRIMATEs in Section 4.1.5.

4.1.4 Applications to GFN$_d$-II

In [163], we first prove tighter lower bounds on the number of linearly active S-boxes in CLEFIA-type GFNs with DSM and single-SD function. We show that every 6 rounds of such GFNs provide 50% more linearly active S-boxes than proven previously. Moreover, with the help of MILP, we experimentally demonstrate for the first time that the new bound is tight for up to at least 12 rounds. Thus, our new result suggests that the efficiency of GFN$_d$ with single SD function is equally improved both by moving from single to double SD functions and by going from single-round diffusion to DSM over multiple rounds.

4.1.5 Design PRIMATEs permutation

The contributions of MILP to PRIMATE are threefold: First, we use it to search the optimal offsets for ShiftRows, which results in optimal bounds against differential and linear cryptanalysis. Second, we show that the differential/linear probability of PRIMATE-80 and PRIMATE-120 is upper-bounded by 2^{-100} and 2^{-196}, therefore the standard differential or linear approach will not lead to

a successful attack. Moreover, we also provide bounds for collision producing trails against differential cryptanalysis.

4.1.6 Applications to Rijndael in the related-key model

In [165] we study Rijndael-160/160 and Rijndael-192/192. We first apply MILP technique to search for the best differential trails of the key schedule which have the minimum number of active S-boxes. Using the idea of local collisions we construct 6-round and 8-round related-key rectangle distinguishers for Rijndael-160/160 and Rijndael-192/192 respectively. Based on the rectangle distinguishers, we give the best attacks on Rijndael-160/160 and Rijndael-192/192 in terms of the attacked rounds.

4.2 ID, ZC, integral cryptanalysis and their links

We describe our impossible-differential cryptanalysis of Rijndael-224 and Rijndael-256 in Section 4.2.1. The integral attack and zero-correlation linear attack are applied to SIMON in Section 4.2.2. We discuss the links among impossible-differential, zero-correlation linear and integral cryptanalysis in Section 4.2.3.

4.2.1 Impossible-differential attacks on Rijndael

Our impossible-differential cryptanalysis of Rijndael-224 and Rijndael-256 is published in [164]. First we find new impossible-differentials which have more active bytes at the end (the previous distinguishers have only one byte while ours have three); with the help of these new differentials the cost of the subkey guesses during the key recovery phase can be reduced. For 9-round Rijndael-224 and Rijndael-256, we substantially reduce the complexities. Moreover, we even cryptanalyse 10 out of 14 rounds for Rijndael-256. Up until the time of writing, our results are the best known attack on both Rijndael-224 and Rijndael-256.

4.2.2 Integral and zero-correlation attack on SIMON

In 2013, NSA proposed two families of highly-optimized block ciphers SIMON and SPECK [15] which provide excellent performance in hardware and software respectively. Moreover both families offer a large number of block sizes and key

sizes such that the users can easily match the security requirements of their application without sacrificing the performance. However, no cryptanalysis results are included in the specification of these algorithms.

There are three lines of cryptanalysis of SIMON. One is dedicating great efforts to the search of the best differentials or linear hulls, based on which the attacker might mount the best differential or linear cryptanalysis of SIMON. For instance, Biryukov et al. propose the threshold search technique [34] and find the best 12-round differential for SIMON32, based on which they give the best differential attack on 19 out of 32 rounds for SIMON32 (in 2014). The other line is focusing on key recovery techniques, based on the known best differentials or linear hulls. One example is [162], where the dynamic key-guessing technique is applied, based on the same differential as in [34]. Impossible-differential cryptanalysis has also been applied to all the versions of SIMON family. Since it is easy to construct an impossible-differential by using the idea of miss-in-the-middle for SIMON family, we treat the work [48] as belonging to the second line, however the impossible-differential cryptanalysis cannot surpass differential or linear cryptanalysis in terms of the attacked rounds, even if some dedicated tricks are used to optimize the complexity of the attacks. Another direction is the cube attack [9]. We don't discuss fault attacks or side channel attack in this thesis.

Integral attack on SIMON

In [166], we apply the integral attack to SIMON. We firstly apply integral cryptanalysis on SIMON32. Because the block size is only 32 bits, a 15-round integral distinguisher with 31 active bits in the input can be found experimentally, based on which we present a key recovery attack on 21 out of 32 rounds for SIMON32, while the previous best results only achieved 19 rounds. Our experiments also show that the number of distinguished rounds rapidly increases when the number of active bits becomes close to the block size. As exploiting integral distinguishers with a large number of active bits in the plaintext is hard in general, our approach is not effective for the larger version including SIMON48. However, according to the experimental results for SIMON32, we may expect that there exist good integral distinguishers of larger versions when the number of active bits is near the block size.

Zero-correlation attacks on SIMON

Zero-correlation attacks are promising approaches to attack more rounds of some block ciphers. The results can be improved by applying the techniques proposed by Bogdanov and Wang [44] to reduce the data complexity. In [166] we

apply the zero-correlation attack to SIMON. Even if our results for SIMON32 and SIMON48 are not better than the best differential or integral attack, with the help of divide-and-conquer technique, they are still better than the best impossible-differential attacks published at Asiacrypt 2014 [48].

4.2.3 The links among ID, ZC and integral cryptanalysis

Although relations between cryptanalytic approaches including differential and linear cryptanalysis, statistical saturation and multidimensional linear cryptanalysis, integral and zero-correlation linear cryptanalysis, impossible-differential and zero-correlation linear cryptanalysis have been investigated [36–39,44,53,106], the link between impossible-differential and integral cryptanalysis has been missing. The motivation of our paper [155] is to bridge this gap and establish links between impossible-differential cryptanalysis and integral cryptanalysis.

Firstly, by introducing the concept of structure and dual structure, we prove that $a \to b$ is an impossible-differential of a structure $\mathcal{E}$ if and only if it is a zero-correlation linear hull of the dual structure $\mathcal{E}^{\perp}$. Secondly, by establishing some Boolean equations, we show that a zero-correlation linear hull always indicates the existence of an integral distinguisher. With this observation, we improve the integral cryptanalysis of several Feistel structures: For CAST-256, we find a 24-round integral distinguisher and give a 28 out of 48 rounds attack which reduce the complexities of the attack in [44]; for SMS4, we improve the integral distinguisher from 10 to 12 rounds; and for Camellia, by constructing an 8-round zero-correlation linear hull, we find a 8-round integral distinguisher. Finally, we conclude that an r-round impossible-differential of $\mathcal{E}$ always leads to an r-round integral distinguisher of the dual structure $\mathcal{E}^{\perp}$. If $\mathcal{E}$ and $\mathcal{E}^{\perp}$ are linearly equivalent, we derive a direct link between impossible-differentials and integral distinguishers of $\mathcal{E}$.

Our results may help to classify different cryptanalytic tools and may facilitate the task of evaluating the security of block ciphers against several attacks.

4.3 Interpolation attack of LowMC

LowMC is a collection of block cipher families introduced at Eurocrypt 2015 by Albrecht et al. [10]. The design is optimized for instantiations of multi-party computation, fully homomorphic encryption, and zero-knowledge proofs where the linear operation is essentially free, while the non-linear operation is expensive.

With respect to three metrics: ANDs/bit, ANDdepth and number of ANDs, LowMC beats all the existing block ciphers. In terms of implementations, when encrypting larger amounts of data in the MPC setting, LowMC can achieve improvements in computation and communication complexity by up to a factor of 5 compared to AES-128.

A unique feature of LowMC is that its internal affine layers are chosen at random, and thus each block cipher family contains a huge number of instances. However for every instance, the only nonlinear operation in the round function of LowMC is the partial S-box layer, in particular the algebraic degree of each 3-bit S-box is 2, this results in a slow diffusion and motivated us to analyse the security of LowMC.

The design proposes two specific block cipher families of LowMC, having 80-bit and 128-bit keys. In our work [68], we mount interpolation attacks (cf. Section 3.4.1) on LowMC, and show that a practically significant fraction of 2^{-38} of its 80-bit key instances can be broken 2^{23} times faster than exhaustive search. Moreover, essentially all instances that are claimed to provide 128-bit security can be broken about 1000 times faster. In order to obtain these results we optimize the interpolation attack using several new techniques. In particular, we present an algorithm that combines two main variants of the interpolation attack, which results in an attack that is more efficient than each of them.

4.4 Conclusions

This chapter presented the main contributions of this thesis. They can be divided into three parts: First, we applied mixed integer linear programming technique to several cases of symmetric-key primitives including block ciphers, stream ciphers and authenticated encryption. Second, we contributed to the cryptanalysis of several block ciphers with Feistel and Substitution-Permutation structures. Finally, we studied the links among impossible-differential, zero-correlation and integral cryptanalysis approaches.

Chapter 5

Conclusion and future work

Block ciphers are fundamental components of symmetric-key cryptography. Therefore this dissertation focuses on the analysis of block ciphers to improve the knowledge on block ciphers and authenticated encryption.

5.1 MILP and its applications

Mixed integer linear programming (MILP) is a frequently used method in business and economics to solve optimization problems. This thesis shows how the MILP technique can be applied in symmetric cryptology for both analysis and design. Differential and linear cryptanalysis are two of the most powerful techniques to analyse symmetric-key primitives. For modern ciphers, resistance against these attacks is therefore a mandatory design criterion. We use MILP to prove security bounds against both differential and linear cryptanalysis. The objective function of the MILP problem is the number of linearly or differentially active S-boxes that we want to minimize. Therefore the only requirement for our technique is that the cipher is composed of a combination of S-box operations, linear permutation layers and/or XOR operations. Consequently, our technique can be applied to a wide variety of ciphers, including block ciphers with generalised Feistel network (GFN) and Substitution-Permutation network (SPN) structures, stream ciphers, the underlying primitives of authenticated encryption (AE) and hash functions which have GFN and SPN structures, as long as they are following the above design criteria. We would like to point out that only little programming is required to obtain this result. A minimally experienced programmer can modify the reference implementation of a cipher,

in order to generate the required MILP problem. In the case of stream cipher Enocoro-128v2, it takes CPLEX less than one minute on a 24-core Intel Xeon X5670 processor to prove security against differential cryptanalysis, and less than four minutes to prove security against linear cryptanalysis. Our technique significantly reduces the workload of designers and cryptanalysts.

Also we note that our technique can be applied to search for the differentials and linear hulls for ciphers as described above. Some of our ongoing works and [156] are such examples. Based on our technique, designers can provide more accurate security margins, and the cryptanalysts might mount more powerful attacks.

The Addition Rotation and XOR (ARX) structures, which have been very popular recently, are used for designing stream ciphers such as Salsa20 [17] and ChaCha [16], and block ciphers such as TEA, XTEA or HIGHT. In particular, two of the five SHA-3 finalists, BLAKE and Skein, follow this design strategy. When applying our technique to ARX structures, the nonlinear operation addition will significantly increase the number of variables of MILP, which might be a challenge to the memory of the off-the-shelf optimization package, especially when we aim at the differentials or linear hulls rather than security bounds.

5.2 Cryptanalysis of block ciphers

We contribute to the cryptanalysis of block ciphers: Our impossible-differential cryptanalysis results of Rijndael-224 and Rijndael-256 are the best results till now. When constructing impossible-differentials for ciphers, usually fewer active words are preferred at the end to avoid a large number of subkey guesses during the key recovery. However, due to the property of the Maximum Distance Separable (MDS) matrix of AES/Rijndael, multiple active bytes can help to reduce the number of subkey guesses and the time complexity of the attack.

Moreover, we analyse National Security Agency's (NSA) recent lightweight block cipher design SIMON by applying impossible-differential, integral and zero-correlation linear cryptanalysis. Among these, we point out that the integral attack is usually powerful on word-oriented structures: for example, it achieves the best attack on AES. However, when moving to bit-oriented ciphers the situation changes: even if the integral attack requires much less chosen plaintexts than differential cryptanalysis, it is harder to extend it to more rounds beyond a certain point. Therefore it might be limited in the number of rounds it can attack. However for SIMON32, the integral distinguisher we find achieves two more rounds than the best known differential, which gives us new insight on the power of integral attacks.

Finally, we successfully analyse a very recent block cipher LowMC [10], which is designed for Multi-Party Computation (MPC) and Fully Homomorphic Encryption (FHE) and has been published at Eurocrypt 2015. We refute the designers' security claim. Our optimized interpolation attack can be applied to additional ciphers with lower algebraic degree.

5.3 Links among cryptanalysis methods

We contribute to the research of links among statistic cryptanalysis methods by deriving the link between impossible-differential and integral cryptanalysis. Our results not only allow to achieve a better understanding of impossible-differential, integral and zero-correlation linear cryptanalysis, but also provide some new insights with respect to these cryptanalytic approaches. Our results help to prove the security of block ciphers against impossible-differential cryptanalysis. Moreover we prove that the zero-correlation linear hull always implies the existence of an integral distinguisher. This proof also provides a novel way for constructing integral distinguisher of block ciphers, by which we construct better integral distinguishers for several Feistel structures.

5.4 Directions for future work

In this section, we conclude with some directions for future research in the cryptanalysis of symmetric-key primitives:

- In [166] we have already shown that for SIMON32 the integral distinguisher is much better than the known differentials/linear hulls in terms of number of rounds it can cover. According to the experimental results for SIMON32, we expect that there exist good integral distinguishers for larger versions when the number of active bits is near the block size. Perhaps it is possible to represent the output as a polynomial of the input, and analyse the algebraic degree of it, to determine the number of active bits in the input of the integral distinguisher.
- The search for differentials of ARX structure has been solved by Leurent [107]. We aim to build an automatical tool for searching for the linear hulls for ARX structures, and apply it to analyse the typical ARX ciphers, for instance, the lightweight block cipher SPECK designed by the NSA.
- When calculating the linear correlation of a linear hull, the effects of the dependent S-boxes need to be carefully considered. Some results [149,150]

on the ePrint discuss this issue for SIMON and KATAN; we notice that they counted some of the linear trails twice, hence the correlation they give might be not correct. We will revise these results.

- There are 29 candidates in the second round of Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR). Cryptanalysis of these algorithms is a natural task.
- The combination of classical cryptanalysis methods is a promising research topic, for example differential-linear cryptanalysis [25, 105] has been widely applied to block ciphers, Message Authentication Code (MAC) algorithms and stream ciphers. It might be interesting to connect two other cryptanalysis methods under some conditions, and apply it to ciphers to obtain better results.

Bibliography

[1] IBM ILOG CPLEX optimizer. http://www.ibm.com/software/integration/optimization/cplex-optimizer/.

[2] Specification of SMS4, block cipher for WLAN products – SMS4 (in Chinese). http://www.oscca.gov.cn/UpFile/200621016423197990.pdf.

[3] The International SAT Competitions. http://www.satcompetition.org/.

[4] ISO/IEC 18033-3:2005. Information technology - Security techniques - Encryption algorithms - Part 3: Block ciphers, 2005.

[5] ISO/IEC 29192-2:2012. Information technology - Security techniques - Lightweight cryptography - Part 2: Block ciphers, 2012.

[6] ISO/IEC 29192-3:2012. Information technology - Security techniques - Lightweight cryptography - Part 3: Stream ciphers, 2012.

[7] C. M. Adams. Constructing symmetric ciphers using the CAST design procedure. *Des. Codes Cryptography*, 12(3):283–316, 1997.

[8] W. W. Adams and P. Loustaunau. *An Introduction to Gröbner Bases*. Graduate Studies in Mathematics. American Mathematical Society, Volume 3, 1994.

[9] Z. Ahmadian, S. Rasoolzadeh, M. Salmasizadeh, and M. R. Aref. Automated dynamic cube attack on block ciphers: Cryptanalysis of SIMON and KATAN. Cryptology ePrint Archive, Report 2015/040, 2015. http://eprint.iacr.org/.

[10] M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner. Ciphers for MPC and FHE. In E. Oswald and M. Fischlin, editors,

Advances in Cryptology - EUROCRYPT 2015, Part I, volume 9056 of *Lecture Notes in Computer Science*, pages 430–454. Springer, 2015.

[11] E. Andreeva, B. Bilgin, A. Bogdanov, A. Luykx, F. Mendel, B. Mennink, N. Mouha, Q. Wang, and K. Yasuda. PRIMATEs v1. CAESAR submission (2014). `http://primates.ae/`.

[12] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita. Camellia: A 128-bit block cipher suitable for multiple platforms - design and analysis. In D. R. Stinson and S. E. Tavares, editors, *Selected Areas in Cryptography - SAC 2000*, volume 2012 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2000.

[13] J.-P. Aumasson, L. Henzen, W. Meier, and R. C.-W. Phan. SHA-3 proposal BLAKE. Submission to NIST (Round 3), 2010. `http://131002.net/blake/blake.pdf`.

[14] P. S. L. M. Barreto and V. Rijmen. The Whirlpool hashing function, 2003. `http://www.larc.usp.br/~pbarreto/whirlpool.zip`.

[15] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404, 2013.

[16] D. J. Bernstein. ChaCha, a variant of Salsa20. In: SASC 2008 – The State of the Art of Stream Ciphers. ECRYPT (2008). `http://cr.yp.to/rumba20.html`.

[17] D. J. Bernstein. The Salsa20 family of stream ciphers. In M. J. B. Robshaw and O. Billet, editors, *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 84–97. Springer, 2008.

[18] D. J. Bernstein. CAESAR, 2012. `http://competitions.cr.yp.to/caesar.html`.

[19] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Sponge functions. ECRYPT Hash Function Workshop, 2007.

[20] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Keccak sponge function family main document (version 2.1). Submission to NIST (Round 2), 2010. `http://keccak.noekeon.org/Keccak-main-2.1.pdf`.

[21] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Duplexing the sponge: Single-pass authenticated encryption and other applications. In A. Miri and S. Vaudenay, editors, *Selected Areas in Cryptography - SAC*

2011, volume 7118 of *Lecture Notes in Computer Science*, pages 320–337. Springer, 2011.

[22] E. Biham. New types of cryptanalytic attacks using related keys. *J. Cryptology*, 7(4):229–246, 1994.

[23] E. Biham, R. J. Anderson, and L. R. Knudsen. Serpent: A new block cipher proposal. In S. Vaudenay, editor, *Fast Software Encryption, FSE 1998*, volume 1372 of *Lecture Notes in Computer Science*, pages 222–238. Springer, 1998.

[24] E. Biham, A. Biryukov, and A. Shamir. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. *J. Cryptology*, 18(4):291–311, 2005.

[25] E. Biham, O. Dunkelman, and N. Keller. Enhancing differential-linear cryptanalysis. In Y. Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 254–266. Springer, 2002.

[26] E. Biham, O. Dunkelman, and N. Keller. New results on boomerang and rectangle attacks. In J. Daemen and V. Rijmen, editors, *Fast Software Encryption - FSE 2002*, volume 2365 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2002.

[27] E. Biham, O. Dunkelman, and N. Keller. Related-key boomerang and rectangle attacks. In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 507–525. Springer, 2005.

[28] E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. In A. Menezes and S. A. Vanstone, editors, *Advances in Cryptology - CRYPTO 1990*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1990.

[29] E. Biham and A. Shamir. Differential cryptanalysis of the full 16-round DES. In E. F. Brickell, editor, *Advances in Cryptology - CRYPTO 1992*, volume 740 of *Lecture Notes in Computer Science*, pages 487–496. Springer, 1992.

[30] B. Bilgin, A. Bogdanov, M. Knezevic, F. Mendel, and Q. Wang. Fides: Lightweight authenticated cipher with side-channel resistance for constrained hardware. In G. Bertoni and J. Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013*, volume 8086 of *Lecture Notes in Computer Science*, pages 142–158. Springer, 2013.

[31] A. Biryukov, C. De Cannière, and M. Quisquater. On multiple linear approximations. In M. K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2004.

[32] A. Biryukov and D. Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In M. Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2009.

[33] A. Biryukov, D. Khovratovich, and I. Nikolic. Distinguisher and related-key attack on the full AES-256. In S. Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 231–249. Springer, 2009.

[34] A. Biryukov, A. Roy, and V. Velichkov. Differential analysis of block ciphers SIMON and SPECK. In C. Cid and C. Rechberger, editors, *Fast Software Encryption - FSE 2014*, volume 8540 of *Lecture Notes in Computer Science*, pages 546–570. Springer, 2014.

[35] A. Biryukov and A. Shamir. Structural cryptanalysis of SASAS. In B. Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 394–405. Springer, 2001.

[36] C. Blondeau, A. Bogdanov, and M. Wang. On the (in)equivalence of impossible differential and zero-correlation distinguishers for Feistel- and Skipjack-type ciphers. In I. Boureanu, P. Owesarski, and S. Vaudenay, editors, *Applied Cryptography and Network Security - ACNS 2014*, volume 8479 of *Lecture Notes in Computer Science*, pages 271–288. Springer, 2014.

[37] C. Blondeau, G. Leander, and K. Nyberg. Differential-linear cryptanalysis revisited. In C. Cid and C. Rechberger, editors, *Fast Software Encryption - FSE 2014*, volume 8540 of *Lecture Notes in Computer Science*, pages 411–430. Springer, 2014.

[38] C. Blondeau and K. Nyberg. New links between differential and linear cryptanalysis. In T. Johansson and P. Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 388–404. Springer, 2013.

[39] C. Blondeau and K. Nyberg. Links between truncated differential and multidimensional linear properties of block ciphers and underlying attack complexities. In P. Q. Nguyen and E. Oswald, editors, *Advances in*

Cryptology - EUROCRYPT 2014, volume 8441 of *Lecture Notes in Computer Science*, pages 165–182. Springer, 2014.

[40] A. Bogdanov. *Analysis and Design of Block Cipher Constructions*. PhD thesis, Ruhr University Bochum, 2009.

[41] A. Bogdanov. On unbalanced Feistel networks with contracting MDS diffusion. *Des. Codes Cryptography*, 59(1-3):35–58, 2011.

[42] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[43] A. Bogdanov and V. Rijmen. Linear hulls with correlation zero and linear cryptanalysis of block ciphers. Cryptology ePrint Archive, Report 2011/123, 2011. `http://eprint.iacr.org/`.

[44] A. Bogdanov and M. Wang. Zero correlation linear cryptanalysis with reduced data complexity. In A. Canteaut, editor, *Fast Software Encryption - FSE 2012*, volume 7549 of *Lecture Notes in Computer Science*, pages 29–48. Springer, 2012.

[45] J. Borghoff, A. Canteaut, T. Güneysu, E. B. Kavun, M. Knezevic, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, P. Rombouts, S. S. Thomsen, and T. Yalçin. PRINCE - A low-latency block cipher for pervasive computing applications - Extended abstract. In X. Wang and K. Sako, editors, *Advances in Cryptology - ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2012.

[46] J. Borst, L. R. Knudsen, and V. Rijmen. Two attacks on reduced IDEA. In W. Fumy, editor, *Advances in Cryptology - EUROCRYPT 1997*, volume 1233 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 1997.

[47] C. Bouillaguet, P.-A. Fouque, and G. Leurent. Security analysis of SIMD. In A. Biryukov, G. Gong, and D. R. Stinson, editors, *Selected Areas in Cryptography - SAC 2010*, volume 6544 of *Lecture Notes in Computer Science*, pages 351–368. Springer, 2010.

[48] C. Boura, M. Naya-Plasencia, and V. Suder. Scrutinizing and improving impossible differential attacks: Applications to CLEFIA, Camellia, LBlock and Simon. In P. Sarkar and T. Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014*, volume 8873 of *Lecture Notes in Computer Science*, pages 179–199. Springer, 2014.

[49] L. Brown, J. Pieprzyk, and J. Seberry. LOKI - A cryptographic primitive for authentication and secrecy applications. In J. Seberry and J. Pieprzyk, editors, *Advances in Cryptology - AUSCRYPT 1990*, volume 453 of *Lecture Notes in Computer Science*, pages 229–236. Springer, 1990.

[50] C. Burwick, D. Coppersmith, E. D'Avignon, R. Gennaro, S. Halevi, C. Jutla, S. M. M. Jr, L. O'Connor, M. Peyravian, J. Luke, O. M. Peyravian, D. Stafford, and N. Zunic. MARS - A candidate cipher for AES. In *First Advanced Encryption Standard (AES) Conference*, 1998.

[51] C. Carlet, P. Charpin, and V. Zinoviev. Codes, bent functions and permutations suitable for des-like cryptosystems. *Des. Codes Cryptography*, 15(2):125–156, 1998.

[52] F. Chabaud and A. Joux. Differential collisions in SHA-0. In H. Krawczyk, editor, *Advances in Cryptology - CRYPTO 1998*, volume 1462 of *Lecture Notes in Computer Science*, pages 56–71. Springer, 1998.

[53] F. Chabaud and S. Vaudenay. Links between differential and linear cryptanalysis. In A. D. Santis, editor, *Advances in Cryptology - EUROCRYPT 1994*, volume 950 of *Lecture Notes in Computer Science*, pages 356–365. Springer, 1994.

[54] N. Courtois and W. Meier. Algebraic attacks on stream ciphers with linear feedback. In E. Biham, editor, *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer, 2003.

[55] CRYPTREC. Cryptography Research and Evaluation Committees: report. Archive, 2002. `http://www.ipa.go.jp/security/enc/CRYPTREC/index-e.html`.

[56] J. Daemen and C. S. K. Clapp. Fast hashing and stream encryption with PANAMA. In S. Vaudenay, editor, *Fast Software Encryption - FSE 1998*, volume 1372 of *Lecture Notes in Computer Science*, pages 60–74. Springer, 1998.

[57] J. Daemen, L. R. Knudsen, and V. Rijmen. The block cipher Square. In E. Biham, editor, *Fast Software Encryption - FSE 1997*, volume 1267 of *Lecture Notes in Computer Science*, pages 149–165. Springer, 1997.

[58] J. Daemen and V. Rijmen. AES proposal: Rijndael. In *First Advanced Encryption Standard (AES) Conference*, 1998.

[59] J. Daemen and V. Rijmen. AES and the wide trail design strategy. In L. R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002*, volume

2332 of *Lecture Notes in Computer Science*, pages 108–109. Springer, 2002.

[60] J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[61] D. Davies and W. Price. The application of digital signatures based on public key cryptosystems. NPL Report, DNACS 39/80, December 1980.

[62] C. De Cannière, O. Dunkelman, and M. Knezevic. KATAN and KTANTAN - A family of small and efficient hardware-oriented block ciphers. In C. Clavier and K. Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009*, volume 5747 of *LNCS*, pages 272–288. Springer, 2009.

[63] C. De Cannière, F. Mendel, and C. Rechberger. Collisions for 70-step SHA-1: On the full cost of collision search. In C. M. Adams, A. Miri, and M. J. Wiener, editors, *Selected Areas in Cryptography - SAC 2007*, volume 4876 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 2007.

[64] C. De Cannière and B. Preneel. Trivium. In M. J. B. Robshaw and O. Billet, editors, *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 244–266. Springer, 2008.

[65] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) protocol version 1.2. RFC 5246 (Proposed Standard), 2008.

[66] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[67] W. Diffie and G. Ledin (translators). SMS4 encryption algorithm for wireless networks. Cryptology ePrint Archive, Report 2008/329, 2008. `http://eprint.iacr.org/`.

[68] I. Dinur, Y. Liu, W. Meier, and Q. Wang. Optimized interpolation attacks on LowMC. In T. Iwata and J. H. Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 535–560. Springer, 2015.

[69] I. Dinur and A. Shamir. Cube attacks on tweakable black box polynomials. Cryptology ePrint Archive, Report 2008/385, 2008. `http://eprint.iacr.org/`.

[70] H. Dobbertin. Cryptanalysis of MD4. *J. Cryptology*, 11(4):253–271, 1998.

[71] M. J. Dworkin. SP 800-38C. Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. Technical report. National Institute of Standards and Technology, Gaithersburg, MD, United States, 2004.

[72] M. J. Dworkin. SP 800-38D. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. Technical report, National Institute of Standards and Technology, Gaithersburg, MD, United States, 2007.

[73] ECRYPT. The eSTREAM project. `http://www.ecrypt.eu.org/stream/`.

[74] N. Ferguson. Authentication weaknesses in GCM (May 20, 2005). `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/Ferguson2.pdf`.

[75] N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno, J. Callas, and J. Walker. The Skein hash function family. Submission to NIST (Round 3), 2010. `http://www.skein-hash.info/sites/default/files/skein1.3.pdf`.

[76] E. Fleischmann, C. Forler, and S. Lucks. McOE: A family of almost foolproof on-line authenticated encryption schemes. In A. Canteaut, editor, *Fast Software Encryption - FSE 2012*, volume 7549 of *Lecture Notes in Computer Science*, pages 196–215. Springer, 2012.

[77] 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 (Round 3), 2011. `http://www.groestl.info/Groestl.pdf`.

[78] H. Gilbert and M. Minier. A collision attack on 7 rounds of Rijndael. In *AES Candidate Conference*, pages 230–241, 2000.

[79] V. D. Gligor and P. Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. In M. Matsui, editor, *Fast Software Encryption - FSE 2001*, volume 2355 of *Lecture Notes in Computer Science*, pages 92–108. Springer, 2001.

[80] Z. Gong, S. Nikova, and Y. W. Law. KLEIN: A new family of lightweight block ciphers. In A. Juels and C. Paar, editors, *RFID. Security and Privacy - RFIDSec 2011*, volume 7055 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2011.

[81] J. Guo, T. Peyrin, A. Poschmann, and M. J. B. Robshaw. The LED block cipher. In B. Preneel and T. Takagi, editors, *Cryptographic Hardware*

and Embedded Systems - CHES 2011, volume 6917 of *Lecture Notes in Computer Science*, pages 326–341. Springer, 2011.

[82] H. Handschuh and B. Preneel. Key-recovery attacks on universal hash function based MAC algorithms. In D. Wagner, editor, *Advances in Cryptology - CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 144–161. Springer, 2008.

[83] M. Hermelin, J. Y. Cho, and K. Nyberg. Multidimensional extension of Matsui's algorithm 2. In *Fast Software Encryption*, pages 209–227. Springer, 2009.

[84] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, and S. Chee. HIGHT: A new block cipher suitable for low-resource device. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 46–59. Springer, 2006.

[85] T. Iwata and K. Yasuda. BTM: A single-key, inverse-cipher-free mode for deterministic authenticated encryption. In M. J. Jacobson Jr., V. Rijmen, and R. Safavi-Naini, editors, *Selected Areas in Cryptography - SAC 2009*, volume 5867 of *Lecture Notes in Computer Science*, pages 313–330. Springer, 2009.

[86] M. J. B. Robshaw and O. Billet, editors. *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*. Springer, 2008.

[87] T. Jakobsen and L. R. Knudsen. The interpolation attack on block ciphers. In E. Biham, editor, *Fast Software Encryption, 4th International Workshop - FSE 1997*, volume 1267 of *Lecture Notes in Computer Science*, pages 28–40. Springer, 1997.

[88] T. Jakobsen and L. R. Knudsen. Attacks on block ciphers of low algebraic degree. *J. Cryptology*, 14(3):197–210, 2001.

[89] A. Joux. Authentication failures in NIST version of GCM (2006). `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/800-38_Series-Drafts/GCM/Joux_comments.pdf`.

[90] C. S. Jutla. Encryption modes with almost free message integrity. *J. Cryptology*, 21(4):547–578, 2008.

[91] B. S. Kaliski Jr. and M. J. B. Robshaw. Linear cryptanalysis using multiple approximations. In Y. Desmedt, editor, *Advances in Cryptology -*

CRYPTO 1994, volume 839 of *Lecture Notes in Computer Science*, pages 26–39. Springer, 1994.

[92] M. Kanda. Practical security evaluation against differential and linear cryptanalyses for Feistel ciphers with SPN round function. In D. R. Stinson and S. E. Tavares, editors, *Selected Areas in Cryptography - SAC 2000*, volume 2012 of *Lecture Notes in Computer Science*, pages 324–338. Springer, 2000.

[93] E. Kavun, M. Lauridsen, G. Leander, C. Rechberger, P. Schwabe, and T. Yalçın. Prøst v1.1. Submission to the CAESAR competition. `http://proest.compute.dtu.dk/proestv11.pdf`.

[94] J. Kelsey, T. Kohno, and B. Schneier. Amplified boomerang attacks against reduced-round MARS and Serpent. In B. Schneier, editor, *Fast Software Encryption - FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 75–93. Springer, 2000.

[95] J. Kelsey, B. Schneier, and D. Wagner. Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA. In Y. Han, T. Okamoto, and S. Qing, editors, *Information and Communication Security - ICICS 1997*, volume 1334 of *Lecture Notes in Computer Science*, pages 233–246. Springer, 1997.

[96] J. Kim, S. Hong, and B. Preneel. Related-key rectangle attacks on reduced AES-192 and AES-256. In A. Biryukov, editor, *Fast Software Encryption - FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 225–241. Springer, 2007.

[97] L. R. Knudsen. Truncated and higher order differentials. In B. Preneel, editor, *Fast Software Encryption - FSE 1994*, volume 1008 of *Lecture Notes in Computer Science*, pages 196–211. Springer, 1994.

[98] L. R. Knudsen and M. J. B. Robshaw. *The Block Cipher Companion*. Information Security and Cryptography. Springer, 2011.

[99] L. R. Knudsen and D. Wagner. Integral cryptanalysis. In J. Daemen and V. Rijmen, editors, *Fast Software Encryption - FSE 2002*, volume 2365 of *Lecture Notes in Computer Science*, pages 112–127. Springer, 2002.

[100] T. Krovetz and P. Rogaway. The software performance of authenticated-encryption modes. In A. Joux, editor, *Fast Software Encryption - FSE 2011*, volume 6733 of *Lecture Notes in Computer Science*, pages 306–327. Springer, 2011.

[101] X. Lai. On the design and security of block ciphers. PhD thesis, Swiss Federal Institute of Technology, 1992.

[102] X. Lai. Higher order derivatives and differential cryptanalysis. In *Proc. Symposium on Communication, Coding and Cryptography in honor of J. L. Massey on the occasion of his 60'th birthday*. Kluwer Academic Publisher, 1994.

[103] X. Lai and J. L. Massey. A Proposal for a New Block Encryption Standard. In I. Damgård, editor, *Advances in Cryptology - EUROCRYPT 1990*, volume 473 of *Lecture Notes in Computer Science*, pages 389–404. Springer, 1990.

[104] X. Lai and J. L. Massey. Markov ciphers and differential cryptanalysis. In D. W. Davies, editor, *Advances in Cryptology - EUROCRYPT 1991*, volume 547 of *Lecture Notes in Computer Science*, pages 17–38. Springer, 1991.

[105] S. K. Langford and M. E. Hellman. Differential-linear cryptanalysis. In Y. Desmedt, editor, *Advances in Cryptology - CRYPTO 1994*, volume 839 of *Lecture Notes in Computer Science*, pages 17–25. Springer, 1994.

[106] G. Leander. On linear hulls, statistical saturation attacks, PRESENT and a cryptanalysis of PUFFIN. In K. G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 303–322. Springer, 2011.

[107] G. Leurent. Construction of differential characteristics in ARX designs application to Skein. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology - CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 241–258. Springer, 2013.

[108] C. H. Lim and T. Korkishko. mCrypton - A lightweight block cipher for security of low-cost RFID tags and sensors. In J. Song, T. Kwon, and M. Yung, editors, *Information Security Applications - WISA 2005*, volume 3786 of *Lecture Notes in Computer Science*, pages 243–258. Springer, 2005.

[109] Y. Liu, L. Li, D. Gu, X. Wang, Z. Liu, J. Chen, and W. Li. New observations on impossible differential cryptanalysis of reduced-round Camellia. In A. Canteaut, editor, *Fast Software Encryption - FSE 2012*, volume 7549 of *Lecture Notes in Computer Science*, pages 90–109. Springer, 2012.

[110] J. Lu. A methodology for differential-linear cryptanalysis and its applications - (extended abstract). In A. Canteaut, editor, *Fast Software Encryption - FSE 2012*, volume 7549 of *Lecture Notes in Computer Science*, pages 69–89. Springer, 2012.

[111] J. Lu and J. Kim. Attacking 44 rounds of the SHACAL-2 block cipher using related-key rectangle cryptanalysis. *IEICE Transactions*, 91-A(9):2588–2596, 2008.

[112] S. Lucks. Attacking seven rounds of Rijndael under 192-bit and 256-bit keys. In *AES Candidate Conference*, pages 215–229, 2000.

[113] J. L. Massey. SAFER K-64: A byte-oriented block-ciphering algorithm. In R. J. Anderson, editor, *Fast Software Encryption - FSE 1993*, volume 809 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 1993.

[114] M. Matsui. Linear cryptanalysis method for DES cipher. In T. Helleseth, editor, *Advances in Cryptology - EUROCRYPT 1993*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer, 1993.

[115] M. Matsui. The first experimental cryptanalysis of the Data Encryption Standard. In Y. Desmedt, editor, *Advances in Cryptology - CRYPTO 1994*, volume 839 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 1994.

[116] M. Matsui and A. Yamagishi. A new method for known plaintext attack of FEAL cipher. In R. A. Rueppel, editor, *Advances in Cryptology - EUROCRYPT 1992*, volume 658 of *Lecture Notes in Computer Science*, pages 81–91. Springer, 1992.

[117] C. M. Matyas, S.M. and J. Oseas. Generating strong one-way functions with cryptographic algorithm. *IBM Technology Disclosure Bulletein*, 27(10A):5658–5659, 1985.

[118] F. Mendel, T. Nad, and M. Schläffer. Cryptanalysis of round-reduced HAS-160. In H. Kim, editor, *Information Security and Cryptology - ICISC 2011*, volume 7259 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2011.

[119] F. Mendel, T. Nad, and M. Schläffer. Finding SHA-2 characteristics: Searching through a minefield of contradictions. In D. H. Lee and X. Wang, editors, *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 288–307. Springer, 2011.

[120] F. Mendel, T. Nad, and M. Schläffer. Finding collisions for round-reduced SM3. In E. Dawson, editor, *The Cryptographers' Track at the RSA Conference - CT-RSA 2013*, volume 7779 of *Lecture Notes in Computer Science*, pages 174–188. Springer, 2013.

[121] R. Merkle. Secrecy, authentication and public key systems. Ph.D. thesis, UMI Research Press, 1979.

[122] M. I. Miyaguchi, S. and K. Ohta. New 128-bit hash function. Proceeding of 4th International Joint Workshop on Computer Communications, Tokyo, Japan, July 13-15, 1989.

[123] S. Miyaguchi. The FEAL cipher family. In A. Menezes and S. A. Vanstone, editors, *Advances in Cryptology - CRYPTO 1990*, volume 537 of *Lecture Notes in Computer Science*, pages 627–638. Springer, 1990.

[124] M. Moos. Cryptominisat. `https://github.com/msoos/cryptominisat`.

[125] N. Mouha, Q. Wang, D. Gu, and B. Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In C. Wu, M. Yung, and D. Lin, editors, *Information Security and Cryptology - Inscrypt 2011*, volume 7537 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2011.

[126] National Institute of Standards and Technology. Cryptographic Hash Algorithm Competition. `http://www.nist.gov/hash-competition`.

[127] National Security Agency. Skipjacjk and KEA Algorithm Specifications, 1998. `http://csrc.nist.gov/groups/ST/toolkit/documents/skipjack/skipjack.pdf`.

[128] K. Nyberg. Linear approximation of block ciphers. In A. D. Santis, editor, *Advances in Cryptology - EUROCRYPT 1994*, volume 950 of *Lecture Notes in Computer Science*, pages 439–444. Springer, 1994.

[129] K. Nyberg. Generalized Feistel Networks. In K. Kim and T. Matsumoto, editors, *Advances in Cryptology - ASIACRYPT 1996*, volume 1163 of *Lecture Notes in Computer Science*, pages 91–104. Springer, 1996.

[130] R. C.-W. Phan. Impossible differential cryptanalysis of 7-round Advanced Encryption Standard (AES). *Inf. Process. Lett.*, 91(1):33–38, 2004.

[131] B. Preneel. NESSIE Project. In *Encyclopedia of Cryptography and Security (2nd Ed.)*, pages 831–836, 2011.

[132] B. Preneel, R. Govaerts, and J. Vandewalle. Hash functions based on block ciphers: A synthetic approach. In D. R. Stinson, editor, *Advances in Cryptology - CRYPTO 1993*, volume 773 of *Lecture Notes in Computer Science*, pages 368–378. Springer, 1993.

[133] G. Procter and C. Cid. On weak keys and forgery attacks against polynomial-based MAC schemes. In S. Moriai, editor, *Fast Software Encryption - FSE 2013*, volume 8424 of *Lecture Notes in Computer Science*, pages 287–304. Springer, 2013.

[134] R. Rivest. The MD5 message-digest algorithm. Internet Engineering Task Force (IETF) Request for Comments (RFC) 1321, April 1992.

[135] R. Rivest. The RC4 encryption algorithm. RSA Data Security, Inc., March 12, 1992.

[136] R. L. Rivest. The MD4 message digest algorithm. In A. Menezes and S. A. Vanstone, editors, *Advances in Cryptology - CRYPTO 1990*, volume 537 of *Lecture Notes in Computer Science*, pages 303–311. Springer, 1990.

[137] R. L. Rivest. The RC5 encryption algorithm. In B. Preneel, editor, *Fast Software Encryption - FSE 1994*, volume 1008 of *Lecture Notes in Computer Science*, pages 86–96. Springer, 1994.

[138] R. L. Rivest, M. J. B. Robshaw, R. Sidney, and Y. L. Yin. The RC6 block cipher. In *First Advanced Encryption Standard (AES) Conference*, 1998.

[139] P. Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In P. J. Lee, editor, *Advances in Cryptology - ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 16–31. Springer, 2004.

[140] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: a block-cipher mode of operation for efficient authenticated encryption. In M. K. Reiter and P. Samarati, editors, *ACM Conference on Computer and Communications Security, CCS 2001*, pages 196–205. ACM, 2001.

[141] P. Rogaway and T. Shrimpton. A provable-security treatment of the key-wrap problem. In S. Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 373–390. Springer, 2006.

[142] M. O. Saarinen. Cycling attacks on GCM, GHASH and other polynomial MACs and hashes. In A. Canteaut, editor, *Fast Software Encryption - FSE 2012*, volume 7549 of *Lecture Notes in Computer Science*, pages 216–225. Springer, 2012.

[143] B. Schneier. Description of a new variable-length key, 64-bit block cipher (Blowfish). In R. J. Anderson, editor, *Fast Software Encryption - FSE 1993, Proceedings*, volume 809 of *Lecture Notes in Computer Science*, pages 191–204. Springer, 1993.

[144] B. Schneier and J. Kelsey. Unbalanced Feistel Networks and block cipher design. In D. Gollmann, editor, *Fast Software Encryption - FSE 1996*, volume 1039 of *Lecture Notes in Computer Science*, pages 121–144. Springer, 1996.

[145] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson. Twofish: A 128-bit block cipher. In *First Advanced Encryption Standard (AES) Conference*, 1998.

[146] A. A. Selçuk. On probability of success in linear and differential cryptanalysis. *J. Cryptology*, 21(1):131–147, 2008.

[147] C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423, 1948.

[148] C. E. Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 28:656–715, 1949.

[149] D. Shi, L. Hu, S. Sun, and L. Song. Improved linear (hull) cryptanalysis of round-reduced versions of KATAN. Cryptology ePrint Archive, Report 2015/964, 2015. `http://eprint.iacr.org/`.

[150] D. Shi, L. Hu, S. Sun, L. Song, K. Qiao, and X. Ma. Improved linear (hull) cryptanalysis of round-reduced versions of SIMON. Cryptology ePrint Archive, Report 2014/973, 2014. `http://eprint.iacr.org/`.

[151] K. Shibutani. On the diffusion of Generalized Feistel structures regarding differential and linear cryptanalysis. In A. Biryukov, G. Gong, and D. R. Stinson, editors, *Selected Areas in Cryptography - SAC 2010*, volume 6544 of *Lecture Notes in Computer Science*, pages 211–228. Springer, 2010.

[152] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai. Piccolo: An ultra-lightweight blockcipher. In B. Preneel and T. Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 342–357. Springer, 2011.

[153] T. Shirai, K. Shibutani, T. Akishita, S. Moriai, and T. Iwata. The 128-bit blockcipher CLEFIA (extended abstract). In A. Biryukov, editor, *Fast Software Encryption - FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 181–195. Springer, 2007.

[154] F.-X. Standaert, G. Piret, G. Rouvroy, J.-J. Quisquater, and J.-D. Legat. ICEBERG: An involutional cipher efficient for block encryption in reconfigurable hardware. In B. K. Roy and W. Meier, editors, *Fast Software Encryption - FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 279–299. Springer, 2004.

[155] B. Sun, Z. Liu, V. Rijmen, R. Li, L. Cheng, Q. Wang, H. AlKhzaimi, and C. Li. Links among impossible differential, integral and zero correlation linear cryptanalysis. In R. Gennaro and M. J. B. Robshaw, editors,

Advances in Cryptology - CRYPTO 2015, Part I, volume 9215 of *Lecture Notes in Computer Science*, pages 95–115. Springer, 2015.

[156] S. Sun, L. Hu, P. Wang, K. Qiao, X. Ma, and L. Song. Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In P. Sarkar and T. Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 158–178. Springer, 2014.

[157] T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi. TWINE: A lightweight block cipher for multiple platforms. In L. R. Knudsen and H. Wu, editors, *Selected Areas in Cryptography - SAC 2012*, volume 7707 of *Lecture Notes in Computer Science*, pages 339–354. Springer, 2012.

[158] A. Tardy-Corfdir and H. Gilbert. A known plaintext attack of FEAL-4 and FEAL-6. In J. Feigenbaum, editor, *Advances in Cryptology - CRYPTO 1991*, volume 576 of *Lecture Notes in Computer Science*, pages 172–181. Springer, 1991.

[159] Y. Tsunoo, E. Tsujihara, M. Shigeri, T. Saito, T. Suzaki, and H. Kubo. Impossible differential cryptanalysis of CLEFIA. In K. Nyberg, editor, *Fast Software Encryption - FSE 2008*, volume 5086 of *Lecture Notes in Computer Science*, pages 398–411. Springer, 2008.

[160] S. Vaudenay. Security flaws induced by CBC padding - applications to SSL, IPSEC, WTLS ... In L. R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 534–546. Springer, 2002.

[161] D. Wagner. The boomerang attack. In L. R. Knudsen, editor, *Fast Software Encryption - FSE 1999*, volume 1636 of *LNCS*, pages 156–170. Springer, 1999.

[162] N. Wang, X. Wang, K. Jia, and J. Zhao. Differential attacks on reduced SIMON versions with dynamic key-guessing techniques. Cryptology ePrint Archive, Report 2014/448, 2014. `http://eprint.iacr.org/`.

[163] Q. Wang and A. Bogdanov. The provable constructive effect of diffusion switching mechanism in CLEFIA-type block ciphers. *Inf. Process. Lett.*, 112(11):427–432, 2012.

[164] Q. Wang, D. Gu, V. Rijmen, Y. Liu, J. Chen, and A. Bogdanov. Improved impossible differential attacks on large-block Rijndael. In T. Kwon, M. Lee, and D. Kwon, editors, *Information Security and Cryptology - ICISC 2012*, volume 7839 of *Lecture Notes in Computer Science*, pages 126–140. Springer, 2012.

[165] Q. Wang, Z. Liu, D. Toz, K. Varici, and D. Gu. Related-key rectangle cryptanalysis of Rijndael-160 and Rijndael-192. *IET Information Security*, 9(5):266–276, 2015.

[166] Q. Wang, Z. Liu, K. Varici, Y. Sasaki, V. Rijmen, and Y. Todo. Cryptanalysis of reduced-round SIMON32 and SIMON48. In W. Meier and D. Mukhopadhyay, editors, *Progress in Cryptology - INDOCRYPT 2014*, volume 8885 of *Lecture Notes in Computer Science*, pages 143–160. Springer, 2014.

[167] X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu. Cryptanalysis of the hash functions MD4 and RIPEMD. In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2005.

[168] X. Wang and H. Yu. How to break MD5 and other hash functions. In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2005.

[169] D. Watanabe, K. Okamoto, and T. Kaneko. A hardware-oriented light weight pseudo-random number generator Enocoro-128v2. In *The Symposium on Cryptography and Information Security*, pages 3D1–3, 2010. (in Japanese).

[170] D. J. Wheeler and R. M. Needham. TEA, A tiny encryption algorithm. In B. Preneel, editor, *Fast Software Encryption - FSE 1994*, volume 1008 of *Lecture Notes in Computer Science*, pages 363–366. Springer, 1994.

[171] H. Wu. The hash function JH. Submission to NIST, 2008. `http://icsd.i2r.a-star.edu.sg/staff/hongjun/jh/jh_round2.pdf`.

[172] S. Wu and M. Wang. Security evaluation against differential cryptanalysis for block cipher structures. Cryptology ePrint Archive, Report 2011/551, 2011. `http://eprint.iacr.org/`.

[173] W. Wu and L. Zhang. LBlock: A lightweight block cipher. In J. Lopez and G. Tsudik, editors, *Applied Cryptography and Network Security - ACNS 2011*, volume 6715 of *Lecture Notes in Computer Science*, pages 327–344, 2011.

[174] H. Yap, K. Khoo, A. Poschmann, and M. Henricksen. EPCBC - A block cipher suitable for Electronic Product Code encryption. In D. Lin, G. Tsudik, and X. Wang, editors, *Cryptology and Network Security - CANS 2011*, volume 7092 of *Lecture Notes in Computer Science*, pages 76–97. Springer, 2011.

Part II

Publications

List of publications

International Journals

1. Y. Liu, A. Yang, Z. Liu, W. Li, Q. Wang, L. Song, D. Gu: Improved Impossible Differential Attack on Reduced Version of Camellia with FL/FL^{-1} Functions. *IET Information Security* 2016. (accepted)

2. Z. Liu, B. Sun, Q. Wang, K. Varıcı, D.Gu: Improved Zero-Correlation Linear Cryptanalysis of Reduced-round Camellia under Weak Keys. *IET Information Security* 10(2): 95-103, 2016.

3. Q. Wang, Z. Liu, D. Toz, K. Varıcı, D. Gu: Related-Key Rectangle Cryptanalysis of Rijndael-160 and Rijndael-192. *IET Information Security* 9(5): 266–276, 2015.

4. Z. Liu, Y. Liu, Q. Wang, D. Gu, W. Li: Meet-in-the-middle fault analysis on word-oriented substitution-permutation network block ciphers. *Security and Communication Networks* 8(4): 672-681, 2015.

5. W. Li, D. Gu, X. Xia, C. Zhao, Z. Liu, Y. Liu, Q. Wang: Single Byte Differential Fault Analysis on the LED Lightweight Cipher in the Wireless Sensor Network. *International Journal of Computational Intelligence Systems* 5(5): 896-904, 2012.

6. Q. Wang, A. Bogdanov: The provable constructive effect of diffusion switching mechanism in CLEFIA-type block ciphers. *Information Processing Letters* 112(11): 427-432, 2012.

International Conferences

1. Y. Liu, Q. Wang, V. Rijmen: Automatic Search of Linear Trails in ARX with Applications to SPECK and Chaskey. In M. Manulis, A-R Sadeghi and S. Schneider, Eds.: Applied Cryptography and Network Security - ACNS 2016, volumn 9696 of *Lecture Notes in Computer Science*, pages 485-499, Springer 2016,
2. I. Dinur, Y. Liu, W. Meier, Q. Wang: Optimized Interpolation Attacks on LowMC. In T. Iwata and J-H Cheon Eds.: Advances in Cryptology - ASIACRYPT 2015, Part II, volumn 9453 of *Lecture Notes in Computer Science*, pages 535-560, Springer 2015.
3. B. Sun, Z. Liu, V. Rijmen, R. Li, L. Cheng, Q. Wang, H. Alkhzaimi, C. Li: Links Among Impossible Differential, Integral and Zero Correlation Linear Cryptanalysis. In R. Gennaro and M.J.B. Robshaw Eds.: Advances in Cryptology - CRYPTO 2015, Part I, volumn 9215 of *Lecture Notes in Computer Science*, pages 95–115, Springer 2015.
4. Q. Wang, Z. Liu, K. Varıcı, Y. Sasaki, V. Rijmen, Y. Todo: Cryptanalysis of Reduced-Round SIMON32 and SIMON48. In W. Meier and D. Mukhopadhyay Eds.: Progress in Cryptology - INDOCRYPT 2014, volume 8885 of *Lecture Notes in Computer Science*, pages 143–160, Springer 2014.
5. B. Bilgin, A. Bogdanov, M. Knežević, F. Mendel, Q. Wang: Fides: Lightweight Authenticated Cipher with Side-Channel Resistance for Constrained Hardware. In G. Bertoni and J.-S. Coron Eds.: Cryptographic Hardware and Embedded Systems - CHES 2013, volume 8086 of *Lecture Notes in Computer Science*, pages 142–158, Springer 2013.
6. Q. Wang, D. Gu, V. Rijmen, Y. Liu, J. Chen, A. Bogdanov: Improved Impossible Differential Attacks on Large-Block Rijndael. In. Kwon, M.-K. Lee, and D. Kwon Eds.: Information Security and Cryptology - ICISC 2012, volume 7839 of *Lecture Notes in Computer Science*, pages 126–140, Springer 2013.
7. N. Mouha, Q. Wang, D. Gu, B. Preneel: Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In C.-K. Wu, M. Yung, and D. Lin Eds.: Information Security and Cryptology - Inscrypt 2011, volumn 7537 of *Lecture Notes in Computer Science*, pages 57-76, Springer 2012.

Other Conferences and Workshops

1. E. Andreeva, B. Bilgin, A. Bogdanov, A. Luykx, F. Mendel, B. Mennink, N. Mouha, Q. Wang, K. Yasuda: PRIMATEs v1.02. Round 2 of CAESAR Competition, 2015.

Chapter 6

Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming

Publication Data

N. Mouha, Q. Wang, D. Gu, B. Preneel: Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In C.-K. Wu, M. Yung, and D. Lin (Eds.): *Inscrypt* 2011, volumn 7537 of *Lecture Notes in Computer Science*, pages 57-76, 2012.

Contributions

Major contributor, except for the C code in the Appendix. The workload is 50%.

Differential and Linear Cryptanalysis using Mixed-Integer Linear Programming*

Nicky Mouha[1,3,**], Qingju Wang[1,2,3], Dawu Gu[2], and Bart Preneel[1,3]

[1] Department of Electrical Engineering ESAT/SCD-COSIC,
Katholieke Universiteit Leuven. Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium.
[2] Department of Computer Science and Engineering, Shanghai Jiao Tong University,
Shanghai, China.
[3] Interdisciplinary Institute for BroadBand Technology (IBBT), Belgium.
{Nicky.Mouha,Qingju.Wang}@esat.kuleuven.be

Abstract. Differential and linear cryptanalysis are two of the most powerful techniques to analyze symmetric-key primitives. For modern ciphers, resistance against these attacks is therefore a mandatory design criterion. In this paper, we propose a novel technique to prove security bounds against both differential and linear cryptanalysis. We use mixed-integer linear programming (MILP), a method that is frequently used in business and economics to solve optimization problems. Our technique significantly reduces the workload of designers and cryptanalysts, because it only involves writing out simple equations that are input into an MILP solver. As very little programming is required, both the time spent on cryptanalysis and the possibility of human errors are greatly reduced. Our method is used to analyze Enocoro-128v2, a stream cipher that consists of 96 rounds. We prove that 38 rounds are sufficient for security against differential cryptanalysis, and 61 rounds for security against linear cryptanalysis. We also illustrate our technique by calculating the number of active S-boxes for AES.

Keywords: Differential cryptanalysis, Linear Cryptanalysis, Mixed-Integer Linear Programming, MILP, Enocoro, AES, CPLEX

1 Introduction

Differential cryptanalysis [1] and linear cryptanalysis [19] have shown to be two of the most important techniques in the analysis of symmetric-key cryptographic primitives. For block ciphers, differential cryptanalysis analyzes how input differences in the plaintext lead to output differences in the ciphertext. Linear cryptanalysis studies probabilistic linear relations between plaintext, ciphertext and

* This work was supported in part by the Research Council K.U.Leuven: GOA TENSE, the IAP Program P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and in part by the European Commission through the ICT program under contract ICT-2007-216676 ECRYPT II, and is funded by the National Natural Science Foundation of China (No. 61073150).

** This author is funded by a research grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

key. If a cipher behaves differently from a random cipher for differential or linear cryptanalysis, this can be used to build a distinguisher or even a key-recovery attack.

For stream ciphers, differential cryptanalysis can be used in the context of a resynchronization attack [11]. In one possible setting, the same data is encrypted several times with the same key, but using a different initial value (IV). This is referred to as the standard (non-related-key) model, where the IV value is assumed to be under control of the attacker. An even stronger attack model is the related-key setting, where the same data is encrypted with different IVs *and* different keys. Not only the IV values, but also the differences between the keys are assumed to be under control of the attacker. Similar to differential cryptanalysis, linear cryptanalysis can also be used to attack stream ciphers in both the standard and related-key model. In the case of stream ciphers, linear cryptanalysis amounts to a known-IV attack instead of a chosen-IV attack.

Resistance against linear and differential cryptanalysis is a standard design criterion for new ciphers. For the block cipher AES [13], provable security against linear and differential cryptanalysis follows from the wide trail design strategy [12]. In this work, we apply a similar strategy. After proving a lower bound on the number of active S-boxes for both differential and linear cryptanalysis, we use the maximum differential probability (MDP) of the S-boxes to derive an upper bound for the probability of the best characteristic. We assume (as is commonly done) that the probability of the differential can accurately be estimated by the probability of the best characteristic. Several works focus on calculating the minimum number of active S-boxes for both Substitution-Permutation Networks (SPNs) [12] and (Generalized) Feistel Structures (GFSs) [5, 6, 16, 24]. Unfortunately, it seems that a lot of time and effort in programming is required to apply those techniques. This may explain why many related constructions have not yet been thoroughly analyzed. In this paper, we introduce a novel technique using mixed-integer linear programming in order to overcome these problems.

Linear programming (LP) is the study of optimizing (minimizing or maximizing) a linear objective function $f(x_1, x_2, \ldots, x_n)$, subject to linear inequalities involving decision variables x_i, $1 \leq i \leq n$. For many such optimization problems, it is necessary to restrict certain decision variables to integer values, i.e. for some values of i, we require $x_i \in \mathbb{Z}$. Methods to formulate and solve such programs are called mixed-integer linear programming (MILP). If all decision variables x_i must be integer, the term (pure) integer linear programming (ILP) is used. MILP techniques have found many practical applications in the fields of economy and business, but their application in cryptography has so far been limited. For a good introductory level text on LP and (M)ILP, we refer to Schrage [23].

In [7], Borghoff *et al.* transformed the quadratic equations describing the stream cipher Bivium into a MILP problem. The IBM ILOG CPLEX Optimizer[4] was then used to solve the resulting MILP problem, which corresponds to recovering the internal state of Bivium. In the case of Bivium A, solving this

[4] `http://www.ibm.com/software/integration/optimization/cplex-optimizer/`

MILP problem takes less than 4.5 hours, which is faster than Raddum's approach (about a day) [22], but much slower than using MiniSAT (21 seconds) [9].

For the hash function SIMD, Bouillaguet *et al.* [8] used an ILP solver to find a differential characteristic based on local collisions. Using the SYMPHONY solver[5], they could not find the optimal solution, but found lower bounds for both SIMD-256 and SIMD-512. The computation for SIMD-512 took one month on a dual quad-core computer.

In [5, 6], Bogdanov calculated the minimum number of linearly and differentially active S-boxes of unbalanced Feistel networks with contracting MDS diffusion. He proved that some truncated difference weight distributions are impossible or equivalent to others. For the remaining truncated difference weight distributions, he constructed an ILP program which he then solved using the MAGMA[6] Computational Algebra System [4]. Compared to Bogdanov's technique, the fully automated method in this paper is much simpler to apply: Bogdanov's approach requires a significant amount of manual work, and the construction of not one but several ILP programs. We will show how this can be avoided by introducing extra dummy variables into the MILP program.

While this paper was under submission, Wu and Wang released a paper on ePrint [28] that also uses integer linear programming to count the number of active S-boxes for both linear and differential cryptanalysis. Just as in Bogdanov's approach, their algorithms require a large number of ILP programs to be solved, instead of only one as in the technique of this paper.

We apply our technique to the stream cipher Enocoro-128v2 [26, 27], in order to obtain bounds against differential and linear cryptanalysis. We consider both the standard and related-key model. All MILP programs are solved using CPLEX. There are 96 initialization rounds in Enocoro-128v2. We prove that 38 rounds are sufficient for security against differential cryptanalysis, and 61 rounds against linear cryptanalysis. These security bounds are obtained after 52.68 and 228.94 seconds respectively. We also calculate the minimum number of active S-boxes for up to 14 rounds of AES, which takes at most 0.40 seconds for each optimization program. Our experiments are performed on a 24-core Intel Xeon X5670 Processor, with 16 GB of RAM.

This paper is organized as follows. Sect. 2 explains how to find the minimum number of active S-boxes for a cryptographic primitive by solving an MILP program. A brief description of Enocoro-128v2 is given in Sect. 3. In Sect. 4 and Sect. 5, we construct an MILP program to prove that Enocoro-128v2 is secure against differential cryptanalysis and linear cryptanalysis respectively. We provide some ideas for future work in Sect. 6, and conclude the paper in Sect. 7. In App. A, we calculate the minimum number of active S-boxes for AES using our technique, and provide the full source code of our program.

[5] `http://projects.coin-or.org/SYMPHONY`
[6] `http://magma.maths.usyd.edu.au/`

2 Constructing an MILP Program to Calculate the Minimum Number of Active S-boxes

We now explain a technique to easily prove the security of many ciphers against differential and linear cryptanalysis. Our method is based on counting the minimum number of active S-boxes. To illustrate our technique, we use Enocoro-128v2 and AES as test cases in this paper. The equations we describe are not specific to these ciphers, but can easily be applied to any cipher constructed using S-box operations, linear permutation layers, three-forked branches and/or XOR operations.

2.1 Differential Cryptanalysis

We consider truncated differences, that is, every byte in our analysis can have either a zero or a non-zero difference. More formally, we define the following difference vector:

Definition 1 *Consider a string Δ consisting of n bytes $\Delta = (\Delta_0, \Delta_1, \ldots, \Delta_{n-1})$. Then, the difference vector $x = (x_0, x_1, \ldots, x_{n-1})$ corresponding to Δ is defined as*

$$x_i = \begin{cases} 0 & \text{if } \Delta_i = 0 \ , \\ 1 & \text{otherwise} \ . \end{cases}$$

Equations Describing the XOR Operation. Let the input difference vector for the XOR operation be $(x^{\oplus}_{in_1}, x^{\oplus}_{in_2})$ and the corresponding output difference vector be $x^{\oplus}_{out}$. The differential branch number is defined as the minimum number of input and output bytes that contain differences, excluding the case where there are no differences in inputs nor outputs. For XOR, the differential branch number is 2. In order to express this branch number in equations, we need to introduce a new binary dummy variable $d^{\oplus}$.[7] If and only if all of the three variables $x^{\oplus}_{in_1}$,$x^{\oplus}_{in_2}$ and $x^{\oplus}_{out}$ are zero, $d^{\oplus}$ is zero, otherwise it should be one. Therefore we obtain the following linear equations (in binary variables) to describe the relation between the input and output difference vectors:

$$\begin{aligned} x^{\oplus}_{in_1} + x^{\oplus}_{in_2} + x^{\oplus}_{out} &\geq 2d^{\oplus} \ , \\ d^{\oplus} &\geq x^{\oplus}_{in_1} \ , \\ d^{\oplus} &\geq x^{\oplus}_{in_2} \ , \\ d^{\oplus} &\geq x^{\oplus}_{out} \ . \end{aligned}$$

[7] Note that this extra variable was not added in [5,6], which is why Bogdanov had to solve several ILP programs instead of only one.

Equations Describing the Linear Transformation. The equations for a linear transformation L can be described as follows. Assume L transforms the input difference vector $(x^L_{in_1}, x^L_{in_2}, \cdots, x^L_{in_\mathcal{M}})$ to the output difference vector $(x^L_{out_1}, x^L_{out_2}, \cdots, x^L_{out_\mathcal{M}})$. Given the differential branch number $\mathcal{B}_D$, a binary dummy variable d^L is again needed to describe the relation between the input and output difference vectors. The variable d^L is equal to 0 if all variables $x^L_{in_1}, x^L_{in_2}, \cdots, x^L_{in_\mathcal{M}}, x^L_{out_1}, x^L_{out_2}, \cdots, x^L_{out_\mathcal{M}}$ are 0, and 1 otherwise. Therefore the linear transformation L can be constrained by the following linear equations:

$$\begin{aligned} x^L_{in_1} + x^L_{in_2} + \cdots + x^L_{in_\mathcal{M}} + x^L_{out_1} + x^L_{out_2} + \cdots + x^L_{out_\mathcal{M}} &\geq \mathcal{B}_D d^L \ , \\ d^L &\geq x^L_{in_1} \ , \\ d^L &\geq x^L_{in_2} \ , \\ &\cdots\cdots \\ d^L &\geq x^L_{in_\mathcal{M}} \ , \\ d^L &\geq x^L_{out_1} \ , \\ d^L &\geq x^L_{out_2} \ , \\ &\cdots\cdots \\ d^L &\geq x^L_{out_\mathcal{M}} \ . \end{aligned}$$

The Objective Function. The objective function that has to be minimized, is the number of active S-boxes. This function is equal to the sum of all variables that correspond to the S-box inputs.

Additional Constraints. An extra linear equation is added to ensure that at least one S-box is active: this avoids the trivial solution where the minimum active S-boxes is zero. If all d-variables and all x-variables are restricted to be binary, the resulting program is a pure ILP (Integer Linear Programming) problem. If all d-variables are restricted to be binary, but only the x-variables corresponding to the input (plaintext), the equations ensure that the optimal solution for all other x-variables will be binary as well. This is similar to Borghoff's suggestion in [7], and results in an MILP (Mixed-Integer Linear Programming) problem that may be solved faster.

2.2 Linear Cryptanalysis

For linear cryptanalysis, we define a linear mask vector as follows:

Definition 2 *Given a set of linear masks $\Gamma = (\Gamma_0, \Gamma_1, \ldots, \Gamma_{n-1})$, the linear mask vector $y = (y_0, y_1, \ldots, y_{n-1})$ corresponding to Γ is defined as*

$$y_i = \begin{cases} 0 & \text{if } \Gamma_i = 0 \ , \\ 1 & \text{otherwise} \ . \end{cases}$$

The duality between differential and linear cryptanalysis was already pointed out by Matsui [20]. The equations describing a linear function are the same as in the case for differential cryptanalysis, however the differential branch number $\mathcal{B}_D$ is replaced by the linear branch number $\mathcal{B}_L$. The linear branch number is the minimum number of non-zero linear masks for the input and output of a function, excluding the all-zero case. No extra equations are introduced for the XOR operations, because the input and output linear masks are the same.

For a three-forked branch, we proceed as follows. Let the input linear mask vector for the three-forked branch be $y^{\vdash}_{in}$, and the corresponding output linear mask vector be $(y^{\vdash}_{out_1}, y^{\vdash}_{out_2})$. We introduce a binary dummy variable $l^{\vdash}$ to generate the following linear equations for the three-forked branch:

$$\begin{aligned} y^{\vdash}_{in} + y^{\vdash}_{out_1} + y^{\vdash}_{out_2} &\geq 2l^{\vdash} \ , \\ l^{\vdash} &\geq y^{\vdash}_{in} \ , \\ l^{\vdash} &\geq y^{\vdash}_{out_1} \ , \\ l^{\vdash} &\geq y^{\vdash}_{out_2} \ . \end{aligned}$$

3 Description of Enocoro-128v2

The first Enocoro specification was given in [25]. Enocoro is a stream cipher, inspired by the Panama construction [10]. Two versions of Enocoro were specified: Enocoro-80v1 with a key size of 80 bits, and Enocoro-128v1 with a key size of 128 bits. Later, a new version for the 128-bit key size appeared in [15]. It is referred to as Enocoro-128v1.1. We now give a short description of Enocoro-128v2. For more details, we refer to the design document [26, 27].

Internal state. The internal state of Enocoro-128v2 is composed of a buffer b consisting of 32 bytes $(b_0, b_1, \ldots, b_{31})$ and a state a consisting of two bytes (a_0, a_1). The initial state is loaded with a 128-bit key K and a 64-bit IV I as follows:

$$\begin{aligned} b_i^{(-96)} &= K_i, \qquad 0 \leq i < 16 \ , \\ b_{i+16}^{(-96)} &= I_i, \qquad 0 \leq i < 8 \ . \end{aligned}$$

All other internal state bytes are loaded with predefined constants.

Update Function. The update function $Next$ uses functions ρ and λ to update the internal state as follows:

$$(a^{(t+1)}, b^{(t+1)}) = Next(S^{(t)}) = (\rho(a^{(t)}, b^{(t)}), \lambda(a^{(t)}, b^{(t)})) \ .$$

An schematic overview of this function is given in Fig. 1.

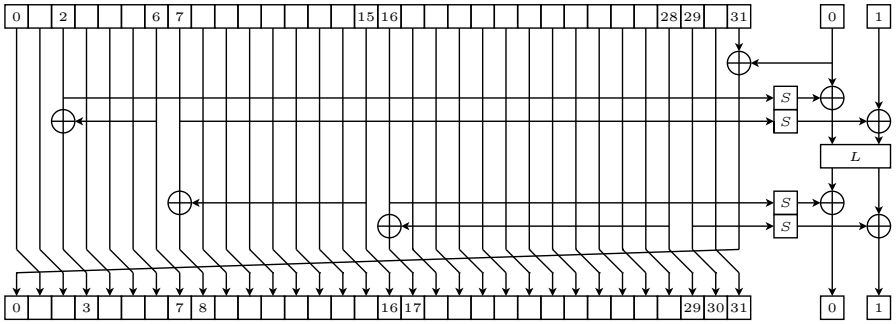


Fig. 1. State Update during the Initialization of Enocoro-128v2. Indices of buffer (on the left) refer to b-variables, indices of the state (on the right) refer to a-variables.

Function ρ. The function ρ updates the state a. It consists of an 8-bit S-box operation, a linear transformation L and XORs. The transformation L is defined as a linear transformation with a 2-by-2 matrix over GF(2^8):

$$\begin{pmatrix} v_0 \\ v_1 \end{pmatrix} = L(u_0, u_1) = \begin{pmatrix} 1 & 1 \\ 1 & d \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \end{pmatrix} \quad , \qquad d \in \mathrm{GF}(2^8) \ ,$$

where $d = \texttt{0x02}$, $u_0 = a_0^{(t)} \oplus S[b_2^{(t)}]$ and $u_1 = a_1^{(t)} \oplus S[b_7^{(t)}]$. The updated state $(a_0^{(t+1)}, a_1^{(t+1)})$ is then calculated as follows:

$$\begin{aligned} a_0^{(t+1)} &= v_0 \oplus S[b_{16}^{(t)}] \ , \\ a_1^{(t+1)} &= v_1 \oplus S[b_{29}^{(t)}] \ . \end{aligned}$$

Function λ. The λ function of Enocoro-128v2 consists of XOR operations and a byte-wise rotation of the buffer b. It is defined as follows:

$$b_i^{(t+1)} = \begin{cases} b_{31}^{(t)} \oplus a_0^{(t)}, & \text{if } i = 0 \ , \\ b_2^{(t)} \oplus b_6^{(t)}, & \text{if } i = 3 \ , \\ b_7^{(t)} \oplus b_{15}^{(t)}, & \text{if } i = 8 \ , \\ b_{16}^{(t)} \oplus b_{28}^{(t)}, & \text{if } i = 17 \ , \\ b_{i-1}^{(t)} & \text{otherwise} \ . \end{cases}$$

Output function *Out*. After 96 initialization rounds, the Enocoro-128v2 output function outputs the lower byte of the state.

$$Out(S^{(t)}) = a_1^{(t)} \ .$$

Several results [14, 17, 18, 21, 27] on differential and linear cryptanalysis have already been published for different versions of Enocoro. In this paper, we consider the most recent version Enocoro-128v2 [26, 27] as an example to illustrate our technique. Watanabe *et al.* already showed that at least $2^{177.8}$ chosen IVs are required for a differential attack on Enocoro-128v2 [27]. For a linear attack, Konosu *et al.* [18] showed that 2^{216} known IVs are required for an attack on the 64-round variant Enocoro-128v1.1. Although these results are already sufficient to prove the security of Enocoro-128v2 against linear and differential cryptanalysis, we explain in this paper how to prove the security against these attacks in a much easier way.

4 Differential Cryptanalysis of Enocoro-128v2

Our technique is now used to find the minimum number of active S-boxes for the stream cipher Enocoro-128v2. We will consider an idealized variant of Enocoro-128v2, for which the minimum number of active S-boxes is a lower bound for the real Enocoro-128v2. In this idealized variant of Enocoro-128v2, the S-boxes can map any non-zero input difference to any non-zero output difference. The same holds for the L-function, with the restriction that the branch number is 3.

For this idealized variant of Enocoro-128v2, we have written a program to calculate the minimum number of active S-boxes. We present our problem as a mixed-integer linear programming (MILP) problem, and use CPLEX to solve it. The solution corresponds to the minimum number of differentially active S-boxes for Enocoro-128v2. It is used to prove the security of the cipher against differential cryptanalysis, using a similar proof as for the block cipher AES [12, 13]. Note that an actual characteristic with the given number of active S-boxes may or may not exist, depending on the specific S-box and L-function that is used. This is not a concern for us, as our goal is to prove a security bound against differential cryptanalysis.

4.1 Constructing the MILP Program

Enocoro-128v2 has eight XOR operations and one linear transformation L in each round. We represent the differential behavior of each of these operations by a set of linear equations, as described in Sect. 2. Let us take the first round of Enocoro-128v2 as an example. The initial difference vector in the buffer and states is represented by the binary variables $(x_0, x_1, \ldots, x_{31})$ and (x_{32}, x_{33}) respectively. Let us consider the XOR operation which has the rightmost byte of buffer b, i.e. b_{31}, and state byte a_0 as inputs. These correspond to binary variables x_{31} and x_{32} respectively, the input difference vector for this XOR operation. From the update function, we can obtain the corresponding value of the leftmost byte of buffer b, i.e. b_0, after the first round. Let the corresponding output difference vector be x_{34}, which is the first new binary variable that we

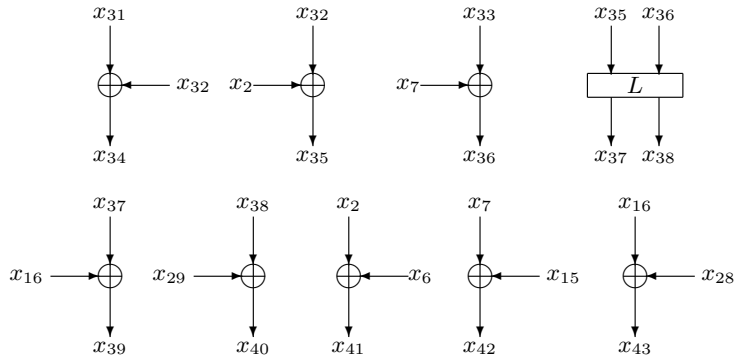


Fig. 2. Difference Vectors for Nine Operations in the First Round

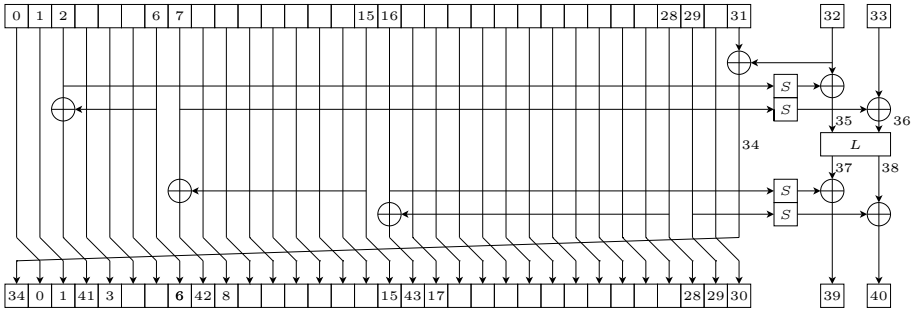


Fig. 3. Differential State Update during the Initialization of Enocoro-128v2. The indices refer to x-variables.

introduce. After introducing a binary dummy variable d_0, this XOR operation can be described by the equations:

$$\begin{aligned} x_{31} + x_{32} + x_{34} &\geq 2d_0 \ , \\ d_0 &\geq x_{31} \ , \\ d_0 &\geq x_{32} \ , \\ d_0 &\geq x_{34} \ . \end{aligned}$$

We now consider the second XOR operation, for which buffer b_2 (input to the first S-box) and the state a_0 are the inputs. Because the S-box is bijective, it is not only the case that the zero input difference results in a zero output difference, but also that a non-zero input difference results in a non-zero output difference. We find that (x_2, x_{32}) is the difference vector of the second XOR operation. The second new variable, x_{35}, will be the output difference vector for

this second XOR operation. Similarly, for the third XOR operation, the input difference vector is (x_7, x_{33}) (corresponding to (b_7, a_1)), and the output difference vector is x_{36}. Given two binary dummy variables d_1 and d_2 for the second and third XOR operation respectively, we again obtain four linear equations for every XOR operation.

From the structure of the linear transformation of Enocoro-128v2, we know that (x_{35}, x_{36}) is the input difference vector for the linear transformation L in the first round. By introducing a new binary variable d_3, the relations between the output difference vector (x_{37}, x_{38}) and the input difference vector (x_{35}, x_{36}) are easily described by the following equations:

$$\begin{aligned} x_{35} + x_{36} + x_{37} + x_{38} &\geq 3d_3 \ , \\ d_3 &\geq x_{35} \ , \\ d_3 &\geq x_{36} \ , \\ d_3 &\geq x_{37} \ , \\ d_3 &\geq x_{38} \ . \end{aligned}$$

The other five XORs in the first round are represented in a similar way. The new variables x_{39}, x_{40}, x_{41}, x_{42} and x_{43} are shown in Fig. 2. These equations result in the binary dummy variables d_4, d_5, d_6, d_7, d_8. For all the eight XORs and one linear transformation L, ten new binary variables $x_{34}, x_{35}, \ldots, x_{43}$ and nine binary dummy variables $d_0, d_1, \ldots, d_8$ are required. Therefore, a system of $4 \cdot 8 + 5 \cdot 1 = 37$ equations is obtained to describe all the nine operations in the first round (and also every subsequent round) of Enocoro-128v2. The detailed input and output vectors for all the nine operations are shown in Fig. 2.

After one round the difference vector for buffer and state will be

$$(x_{34}, x_0, x_1, x_{41}, x_3, \ldots, x_6, x_{42}, x_8, \ldots, x_{15}, x_{43}, x_{17}, \ldots, x_{30})$$

and (x_{39}, x_{40}) respectively. All binary x_i-variables obtained for the first round are illustrated in Fig. 3. Therefore, using this technique we can represent the differential update of Enocoro-128v2 for any round with a system of linear equations.

4.2 The Minimum Number of Active S-boxes for Differential Cryptanalysis

We now focus on the variables that represent the S-box inputs in every round. Note that x_2, x_7, x_{16}, and x_{29} correspond to the input differences of the S-boxes, and therefore determine if the S-box is active or not. Let D_i include the four indices of variables that represent the four S-box inputs in the i-th round $(1 \leq i \leq 96)$. The 96 sets include the indices for variables that represent the four S-box inputs in each round. They can easily be obtained from Sect. 4.1, and are

as follows:

$$\begin{aligned} D_1 &= \{2, 7, 16, 29\} \ , \\ D_2 &= \{1, 6, 15, 28\} \ , \\ D_3 &= \{0, 5, 14, 27\} \ , \\ D_4 &= \{34, 4, 13, 26\} \ , \\ D_5 &= \{44, 3, 12, 25\} \ , \\ &\vdots \\ D_{96} &= \{954, 941, 902, 863\} \ . \end{aligned}$$

Let k_N be the number of active S-boxes for N rounds of Enocoro-128v2. If

$$I_N = \bigcup_{1 \le i \le N} D_i \ ,$$

then

$$k_N = \sum_{i \in I_N} x_i$$

will be the number of active S-boxes in N rounds of Enocoro-128v2. To avoid the trivial case where no S-boxes are active, we add an extra linear constraint to specify that least one S-box is active. If we can minimize the linear function $k_N = \sum_{i \in I_N} x_i$, it will give us the minimum number of active S-boxes for N rounds of Enocoro-128v2. This will provide a security bound for Enocoro-128v2 against differential cryptanalysis. The objective function $k_N = \sum_{i \in I_N} x_i$ is a linear function, constrained by a system of $37N$ linear equations. If all variables must be binary variables, this corresponds to an ILP program.

It is easy to verify that the maximum differential probability for the 8-bit S-box of Enocoro-128v2 is $2^{-4.678}$. As the IV is limited to 64 bits, there are at most 2^{64} IV pairs for any given difference (if the key is fixed). Because there exists a generic attack with a data complexity of 2^{64} IVs (obtaining the entire codebook under one key), attacks requiring 2^{64} IVs or more should not be feasible. Therefore, we do not consider attacks using more than 2^{64} IVs, even in the related-key setting. If the number of active S-boxes in the initialization rounds is at least $14 > 64/4.678$, we consider the cipher to be resistant against differential cryptanalysis. Because we allow differences in both the key and the IV, our results hold both in the single-key and in the related-key setting. We note that typically, differential and linear cryptanalysis are used to attack a few more rounds than the number of rounds of the characteristic. The cipher must also be resistant against other types of attacks and add extra rounds to provide a security margin. For these reasons, more rounds should be used than suggested by our analysis.

In order to optimize the MILP program, we use CPLEX. The experiments are implemented on a 24-core Intel Xeon X5670 @ 2.93 GHz, with 16 GB of RAM. Because this computer is shared with other users, execution times may be longer than necessary, which is why we do not give timing information for all problem

instances. We found that it takes about 52.68 seconds to show that the minimum number of active S-boxes for 38 rounds of Enocoro-128v2 is 14. Therefore, 38 rounds of Enocoro-128v2 or more are secure against differential cryptanalysis. The minimum number of active S-boxes for each round of Enocoro-128v2 are listed in Table 1.

We would like to point out to the reader, that the seemingly complex bookkeeping of variable indices should not be a concern for the cryptanalyst who wishes to use this technique. The MILP linear equations can be generated by a small computer program. This program keeps track of the next unused x- and d-variables. It is then easy to replace every XOR and L function operation in the reference implementation of the cipher by a function to generate the corresponding equations, and every S-box application by a function that constructs the objective function. For a typical cipher, this should not require more than half an hour of work for a minimally experienced programmer.

If all d-variables are restricted to binary variables, as well as variables x_0 up to x_{33}, the equations ensure that the optimal solution for all other x_i-variables will be binary as well. Therefore, similar to Borghoff's suggestion in [7], we might solve an MILP program where only the d-variables and x_0 up to x_{33} are binary variables, instead of a pure ILP program. We find that Borghoff's observation can give dramatic speed-ups in some cases: for 72 rounds, it takes 5,808.15 seconds using an MILP, compared 342,747.78 seconds using a pure ILP. However, our MILP program for 38 rounds takes longer: 75.68 seconds instead of 52.68 seconds. Explaining this phenomenon seems to be a useful direction for future work.

5 Linear Cryptanalysis of Enocoro-128v2

We will use our technique to analyze an ideal variant of Enocoro-128v2 for linear cryptanalysis. Similarly as for differential cryptanalysis, the real Enocoro-128v2 will have at least as many linearly active S-boxes as the idealized one, and therefore can be used to prove a security bound.

5.1 Constructing the MILP Program

We now illustrate our technique by presenting the equations for the first round of the stream cipher Enocoro-128v2 for linear cryptanalysis. For the initial state, let the linear mask vector for the buffer be $(y_0, y_1, \ldots, y_{31})$, and for the state be (y_{32}, y_{33}). Consider the three-forked branch, which has the state byte a_0 as the input linear mask and buffer byte b_{31} as one output linear mask. We obtain the first new binary variable y_{34} as the other output vector. The input and output linear mask vector for this three-forked branch are then y_{32} and (y_{31}, y_{34}) respectively. By introducing the binary dummy variable l_0, the four

Table 1. Minimum Number of Differentially Active S-boxes $\min(k_N)$ for N rounds of Enocoro-128v2

N	$\min(k_N)$
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	1
14	1
15	1
16	1
17	1
18	1
19	1
20	2

N	$\min(k_N)$
21	2
22	3
23	3
24	3
25	4
26	5
27	7
28	8
29	8
30	8
31	8
32	9
33	9
34	10
35	11
36	12
37	13
38	14
39	15
40	15

N	$\min(k_N)$
41	16
42	17
43	18
44	18
45	18
46	19
47	20
48	20
49	21
50	22
51	22
52	22
53	22
54	22
55	22
56	22
57	23
58	23
59	24
60	24

N	$\min(k_N)$
61	25
62	26
63	27
64	27
65	28
66	29
67	30
68	30
69	30
70	31
71	32
72	34
73	35
74	35
75	36
76	37
77	37
78	38
79	38
80	38

N	$\min(k_N)$
81	39
82	39
83	40
84	40
85	40
86	41
87	42
88	43
89	43
90	44
91	44
92	45
93	45
94	46
95	47
96	47

equations describing the three-forked branch can be described as follows:

$$\begin{aligned} y_{31} + y_{32} + y_{34} &\geq 2l_0 \ , \\ l_0 &\geq y_{31} \ , \\ l_0 &\geq y_{32} \ , \\ l_0 &\geq y_{34} \ . \end{aligned}$$

For the XOR operation, the two inputs and the output all have the same linear mask. The bijectiveness of the S-box implies the linear mask at the output will be non-zero if and only if the input mask is non-zero. Therefore, the linear transformation L has an input linear mask vector of (y_{34}, y_{33}), and an output linear mask vector of (y_{35}, y_{36}). Using a new binary dummy variable l_1, the equations describing the L transformation are:

$$\begin{aligned} y_{34} + y_{33} + y_{35} + y_{36} &\geq 3l_1 \ , \\ l_1 &\geq y_{34} \ , \\ l_1 &\geq y_{33} \ , \\ l_1 &\geq y_{35} \ , \\ l_1 &\geq y_{36} \ . \end{aligned}$$

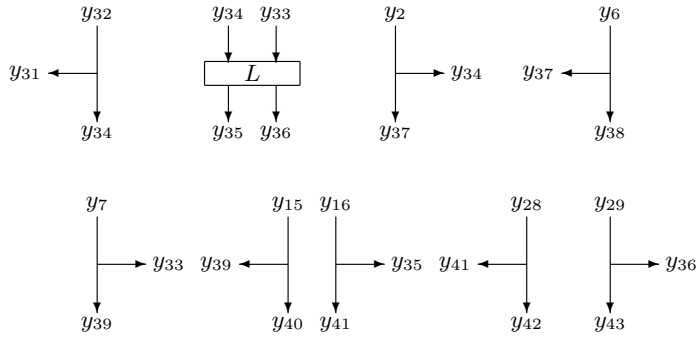


Fig. 4. Linear Mask Vectors for Nine Operations in the First Round

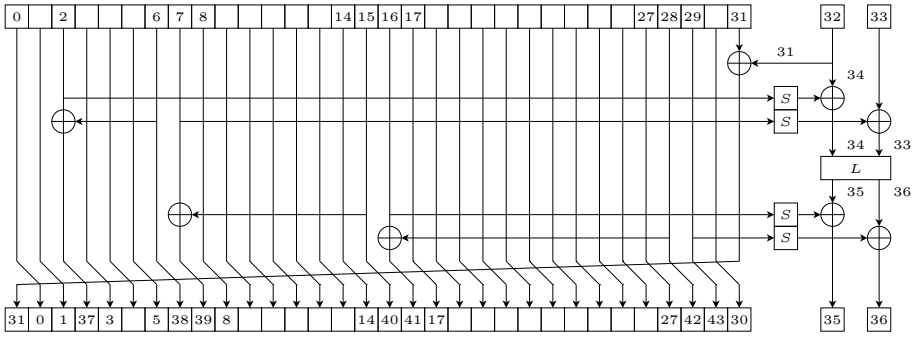


Fig. 5. Linear Mask Vectors Update during the Initialization of Enocoro-128v2. The indices refer to y-variables.

As an Enocoro-128v2 round contains eight three-forked branch operations and one linear transformation L, ten new binary variables $y_{34}, y_{35}, \ldots, y_{43}$, as well as nine binary dummy variables $l_0, l_1, \ldots, l_8$ are introduced. Therefore, $4 \cdot 8 + 5 \cdot 1 = 37$ equations are required to describe the propagation of linear masks for the first round (as well as any subsequent round) of Enocoro-128v2. The input and output linear mask vectors for all nine operations in the first round are shown in Fig. 4. The linear mask vector for the buffer and state after one round are

$$(y_{31}, y_0, y_1, y_{37}, y_3, \cdots, y_5, y_{38}, y_{39}, y_8, \cdots, y_{14}, y_{40}, y_{41}, y_{17}, \cdots, y_{27}, y_{42}, y_{43}, y_{30})$$

and (y_{35}, y_{36}) respectively. They are shown in Fig. 5.

5.2 The Minimum Number of Active S-boxes for Linear Cryptanalysis

Using the technique in the previous section, we can represent any number of rounds of Enocoro-128v2. We now explain how to calculate the number of active S-boxes. Let L_i include all indices of the four variables representing the input linear mask vector of S-boxes in the i-th round ($1 \leq i \leq 96$). We then obtain the following 96 sets:

$$\begin{aligned} L_1 &= \{34, 33, 35, 36\} \ , \\ L_2 &= \{44, 36, 45, 46\} \ , \\ L_3 &= \{54, 46, 55, 56\} \ , \\ L_4 &= \{64, 56, 65, 66\} \ , \\ L_5 &= \{74, 66, 75, 76\} \ , \\ &\vdots \\ L_{96} &= \{984, 976, 985, 986\} \ . \end{aligned}$$

Let m_N be the number of active S-boxes for N rounds of Enocoro-128v2. If

$$J_N = \bigcup_{1 \leq j \leq N} L_j \ ,$$

then

$$m_N = \sum_{j \in J_N} y_j$$

will be the number of active S-boxes for N rounds of Enocoro-128v2. By minimizing the linear objective function m_N, we obtain the minimum number of linearly active S-boxes for N rounds of Enocoro-128v2.

The maximum correlation amplitude of the 8-bit S-box of Enocoro-128v2 is $C_{\max} = 2^{-2}$. For the same reasons as for differential cryptanalysis, we limit the number of IVs to 2^{64}. Let us denote the minimum number of active S-boxes by a. From the limit on the number of IVs, we then find that resistance against linear cryptanalysis requires [13, pp. 142–143]:

$$C_{\max}^a = (2^{-2})^a \leq 2^{-64/2} \ .$$

This inequality is satisfied for $a \geq 16$. Therefore, if the number of linearly active S-boxes is at least 16, Enocoro-128v2 can be considered to be resistant against linear cryptanalysis (in both the single-key and related-key setting).

If we solve the resulting MILP problem using CPLEX, we find that the minimum number of active S-boxes is 18 for 61 rounds of Enocoro-128v2. This result was obtained after 227.38 seconds. Therefore, we conclude that Enocoro-128v2 with 96 initialization rounds is secure against linear cryptanalysis (in both the single-key and related-key setting). The minimum number of active S-boxes for Enocoro-128v2 are listed in Table 2.

Table 2. Minimum Number of Linearly Active S-boxes $\min(m_N)$ for Enocoro-128v2

N	$\min(m_N)$
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	0
17	0
18	0
19	0
20	0

N	$\min(m_N)$
21	0
22	0
23	0
24	0
25	0
26	0
27	0
28	0
29	0
30	0
31	0
32	0
33	3
34	6
35	6
36	6
37	6
38	6
39	6
40	6

N	$\min(m_N)$
41	6
42	9
43	9
44	9
45	12
46	12
47	12
48	12
49	12
50	12
51	12
52	15
53	15
54	15
55	15
56	15
57	15
58	15
59	15
60	15

N	$\min(m_N)$
61	18
62	18
63	18
64	18
65	18
66	18
67	18
68	21
69	21
70	21
71	21
72	21
73	21
74	21
75	21
76	24
77	24
78	24
79	24
80	24

N	$\min(m_N)$
81	24
82	27
83	27
84	27
85	27
86	27
87	27
88	27
89	27
90	27
91	27
92	27
93	30
94	30
95	33
96	33

6 Future Work

It is interesting to investigate how the internal parameters of CPLEX can be fine-tuned to calculate bounds against linear and differential cryptanalysis in the fastest possible time. If there are symmetries in the round function, these may be used to speed up the search as well. Similarly, the attacker may improve a given (suboptimal) lower bound for a particular cipher by clocking the round functions forward or backward in order to obtain a lower number of S-boxes. To obtain a rough lower bound for a large number of rounds, the "split approach" (see for example [3]) may be used. For example, if r rounds of a cipher contain at least a active S-boxes, then kr rounds of a cipher must contain at least ka active S-boxes. It is useful to explore how these observations can be applied when CPLEX takes a very long time to execute. Otherwise, the shorter solving time does not compensate for the additional time to construct the program. For ILP programs with a very long execution time, it may be better to calculate the minimum number of active S-boxes using a different technique (e.g. [3]).

The technique in this paper is quite general, and may also be used for truncated differentials, higher-order differentials, impossible differentials, saturation attacks,... It can also be applied to other ciphers constructed using S-box operations, linear permutation layers, three-forked branches and/or XOR operations. We leave the exploration of these topics to future work as well.

7 Conclusion

In this paper, we introduced a simple technique to calculate the security of many ciphers against linear and differential cryptanalysis. The only requirement is that the cipher is composed of a combination of S-box operations, linear permutation layers and/or XOR operations. Our technique involves writing a simple program to generate a mixed-integer linear programming (MILP) problem. The objective function of the MILP program is the number of linearly or differentially active S-boxes, which we want to minimize. This MILP problem can then easily be solved using an off-the-shelf optimization package, for example CPLEX. The result can be used to prove the security of a cryptosystem against linear and differential cryptanalysis.

Our technique can be applied to a wide variety of cipher constructions. As an example, we apply the technique in this paper to the stream cipher Enocoro-128v2. We prove that for Enocoro-128v2 38 rounds are sufficient for security against differential cryptanalysis, and 61 rounds against linear cryptanalysis. These results are valid both in the single-key and related-key models. As Enocoro-128v2 consists of 96 initialization rounds, this proves the security of Enocoro-128v2 against linear and differential cryptanalysis.

We would like to point out that only little programming is required to obtain this result. A minimally experienced programmer can modify the reference implementation of a cipher, in order to generate the required MILP program in about half an hour. In the case of Enocoro-128v2, it takes CPLEX less than one minute on a 24-core Intel Xeon X5670 processor to prove security against differential cryptanalysis, and less than four minutes to prove security against linear cryptanalysis. We note that because very little programming is required, both the time spent on cryptanalysis and the possibility of making errors are greatly reduced.

Acknowledgments. The authors would like to thank their colleagues at COSIC, as well as the anonymous reviewers for their detailed comments and suggestions. Special thanks to Hirotaka Yoshida for reviewing an earlier draft of this paper.

References

1. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. J. Cryptology 4(1), 3–72 (1991)
2. Biryukov, A., Gong, G., Stinson, D.R. (eds.): Selected Areas in Cryptography - 17th International Workshop, SAC 2010, Waterloo, Ontario, Canada, August 12-13, 2010, Revised Selected Papers, LNCS, vol. 6544. Springer (2011)
3. Biryukov, A., Nikolić, I.: Search for Related-key Differential Characteristics in DES-like ciphers. In: Joux, A. (ed.) FSE. LNCS, vol. 6733, pp. 342–358. Springer (2011)
4. Bodganov, A.: Personal Communication (2011)
5. Bogdanov, A.: Analysis and Design of Block Cipher Constructions. Ph.D. thesis, Ruhr University Bochum (2009)

6. Bogdanov, A.: On unbalanced Feistel networks with contracting MDS diffusion. Des. Codes Cryptography 59(1-3), 35–58 (2011)
7. Borghoff, J., Knudsen, L.R., Stolpe, M.: Bivium as a Mixed-Integer Linear Programming Problem. In: Parker, M.G. (ed.) IMA Int. Conf. LNCS, vol. 5921, pp. 133–152. Springer (2009)
8. Bouillaguet, C., Fouque, P.A., Leurent, G.: Security Analysis of SIMD. In: Biryukov et al. [2], pp. 351–368
9. Cameron McDonald, Chris Charnes, J.P.: An Algebraic Analysis of Trivium Ciphers based on the Boolean Satisfiability Problem. Cryptology ePrint Archive, Report 2007/129 (2007), http://eprint.iacr.org/
10. Daemen, J., Clapp, C.S.K.: Fast Hashing and Stream Encryption with PANAMA. In: Vaudenay, S. (ed.) FSE. LNCS, vol. 1372, pp. 60–74. Springer (1998)
11. Daemen, J., Govaerts, R., Vandewalle, J.: Resynchronization Weaknesses in Synchronous Stream Ciphers. In: EUROCRYPT. pp. 159–167 (1993)
12. Daemen, J., Rijmen, V.: The Wide Trail Design Strategy. In: Honary, B. (ed.) IMA Int. Conf. LNCS, vol. 2260, pp. 222–238. Springer (2001)
13. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer (2002)
14. Hell, M., Johansson, T.: Security Evaluation of Stream Cipher Enocoro-128v2. CRYPTREC Technical Report (2010)
15. K. Muto, D.W., Kaneko, T.: Strength evaluation of Enocoro-128 against LDA and its Improvement. In: Symposium on Cryptography and Information Security. pp. 4A1–1 (2008), (in Japanese)
16. Kanda, M.: Practical Security Evaluation against Differential and Linear Cryptanalyses for Feistel Ciphers with SPN Round Function. In: Stinson, D.R., Tavares, S.E. (eds.) Selected Areas in Cryptography. LNCS, vol. 2012, pp. 324–338. Springer (2000)
17. Kazuto Okamoto, K.M., Kaneko, T.: Security evaluation of Pseudorandom Number Generator Enocoro-80 against Differential/Linear Cryptanalysis (II). In: Symposium on Cryptography and Information Security. pp. 20–23 (2009), (in Japanese)
18. Konosu, K., Muto, K., Furuichi, H., Watanabe, D., Kaneko, T.: Security evaluation of Enocoro-128 ver.1.1 against resynchronization attack. IEICE Technical Report, ISEC2007-147 (2008), (in Japanese)
19. Matsui, M.: Linear Cryptoanalysis Method for DES Cipher. In: EUROCRYPT. pp. 386–397 (1993)
20. Matsui, M.: On Correlation Between the Order of S-boxes and the Strength of DES. In: EUROCRYPT. pp. 366–375 (1994)
21. Muto, K., Watanabe, D., Kaneko, T.: Security evaluation of Enocoro-80 against linear resynchronization attack. Symposium on Cryptography and Information Security (2008), (in Japanese)
22. Raddum, H.: Cryptanalytic Results on Trivium. eSTREAM report 2006/039 (2006), http://www.ecrypt.eu.org/stream/triviump3.html
23. Schrage, L.: Optimization Modeling with LINGO. Lindo Systems (1999), availabe on-line: http://www.lindo.com
24. Shibutani, K.: On the Diffusion of Generalized Feistel Structures Regarding Differential and Linear Cryptanalysis. In: Biryukov et al. [2], pp. 211–228
25. Watanabe, D., Kaneko, T.: A construction of light weight Panama-like keystream generator. In: IEICE Technical Report, ISEC2007-78 (2007), (in Japanese)
26. Watanabe, D., Okamoto, K., Kaneko, T.: A Hardware-Oriented Light Weight Pseudo-Random Number Generator Enocoro-128v2. In: The Symposium on Cryptography and Information Security. pp. 3D1–3 (2010), (in Japanese)

27. Watanabe, D., Owada, T., Okamoto, K., Igarashi, Y., Kaneko, T.: Update on Enocoro Stream Cipher. In: ISITA. pp. 778–783. IEEE (2010)
28. Wu, S., Wang, M.: Security evaluation against differential cryptanalysis for block cipher structures. Cryptology ePrint Archive, Report 2011/551 (2011), `http://eprint.iacr.org/`

A Number of Active S-boxes for AES

The four-round propagation theorem of AES [13] proves that the number of active S-boxes in a differential or linear characteristic of four AES rounds is at least 25. Combined with the properties of the AES S-box, this result was used in the AES design document to prove the resistance against linear and differential attacks. In this section, we illustrate our technique by applying it to the block cipher AES. We not only confirm the four-round propagation theorem, but also determine the minimum number of active S-boxes for up to 14 rounds in Table 4.

An AES round update consists of four operations: AddRoundKey (AR), SubBytes (SB), ShiftRows (SR) and MixColumns (MC). The update of the first AES round is shown in Table 3. Every variable corresponds to a byte of the AES state. The variable is 1 if the difference is non-zero, and 0 if the difference is zero. All variables corresponding to the inputs of the SubByte operations are summed in the objective function, this corresponds to the number of active S-boxes. The linear function used in the MixColumns operation has a differential as well as a linear branch number of 5.

A program was written in C to generate the equations for this optimization problem in the CPLEX LP format. To illustrate the simplicity of our technique, we provide this program (including source code comments) below in full. None of the optimization problems in Table 4 took longer than 0.40 seconds to solve, using only a single core of our 24-core Intel Xeon X5670 processor.

Table 3. The Variables in the First Round Update of AES

$$\begin{bmatrix} x_0 & x_4 & x_8 & x_{12} \\ x_1 & x_5 & x_9 & x_{13} \\ x_2 & x_6 & x_{10} & x_{14} \\ x_3 & x_7 & x_{11} & x_{15} \end{bmatrix} \xrightarrow{\text{SB}} \begin{bmatrix} x_0 & x_4 & x_8 & x_{12} \\ x_1 & x_5 & x_9 & x_{13} \\ x_2 & x_6 & x_{10} & x_{14} \\ x_3 & x_7 & x_{11} & x_{15} \end{bmatrix} \xrightarrow{\text{SR}} \begin{bmatrix} x_0 & x_4 & x_8 & x_{12} \\ x_5 & x_9 & x_{13} & x_1 \\ x_{10} & x_{14} & x_2 & x_6 \\ x_{15} & x_3 & x_7 & x_{11} \end{bmatrix} \xrightarrow{\text{MC}} \begin{bmatrix} x_{16} & x_{20} & x_{24} & x_{28} \\ x_{17} & x_{21} & x_{25} & x_{29} \\ x_{18} & x_{22} & x_{26} & x_{30} \\ x_{19} & x_{23} & x_{27} & x_{31} \end{bmatrix}$$

Table 4. Minimum Number of Differentially or Linearly Active S-boxes $\min(k_N)$ for N rounds of AES

N	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\min(k_N)$	1	5	9	25	26	30	34	50	51	55	59	75	76	80

```
#include <stdio.h>
int i,j,r;
const int ROUNDS = 4; /* number of rounds */
int next = 0; /* next unused state variable index */
int dummy = 0; /* next unused dummy variable index */

void ShiftRows(int a[4][4]) {
  int tmp[4];
  for(i = 1; i < 4; i++) {
    for(j = 0; j < 4; j++) tmp[j] = a[i][(j + i) % 4];
    for(j = 0; j < 4; j++) a[i][j] = tmp[j];
  }
}
void MixColumn(int a[4][4]) {
  for(j = 0; j < 4; j++) {
    for (i = 0; i < 4; i++) printf("x%i + ",a[i][j]);
    for (i = 0; i < 3; i++) printf("x%i + ",next+i);
    printf("x%i - 5 d%i >= 0\n",next+3,dummy);

    for(i = 0; i < 4; i++)
      printf("d%i - x%i >= 0\n",dummy,a[i][j]);
    for(i = 0; i < 4; i++)
      printf("d%i - x%i >= 0\n",dummy,a[i][j]=next++);
    dummy++;
  }
}
int main() {
  int a[4][4]; /* the bytes of the AES state */
  for (i = 0; i < 4; i++)
    for (j = 0; j < 4; j++)
      a[i][j] = next++; /* initialize variable indices */

  printf("Minimize\n"); /* print objective function */
  for (i = 0; i < ROUNDS*16-1; i++) printf("x%i + ",i);
  printf("x%i\n\n",ROUNDS*16-1);

  printf("Subject To\n"); /* round function constraints */
  for (r = 0; r<ROUNDS; r++) { ShiftRows(a); MixColumn(a); }

  /* at least one S-box must be active */
  for (i = 0; i < ROUNDS*16-1; i++) printf("x%i + ",i);
  printf("x%i >= 1\n\n",ROUNDS*16-1);

  printf("Binary\n"); /* binary constraints */
  for (i = 0; i < 16; i++) printf("x%i\n",i);
  for (i = 0; i < dummy; i++) printf("d%i\n",i);
  printf ("End\n");
  return 0;
}
```

Chapter 7

The provable constructive effect of diffusion switching mechanism in CLEFIA-type block ciphers

Publication Data

Q. Wang, A. Bogdanov: The Provable Constructive Effect of Diffusion Switching Mechanism in CLEFIA-type Block Ciphers. *Inf. Process. Lett.* 112(11): 427-432, 2012.

Contributions

Major contributor:

- Get the theorem in Section 3 and prove it.
- Do the experiments in Section 4.

The Provable Constructive Effect of Diffusion Switching Mechanism in CLEFIA-type Block Ciphers

Qingju Wang[1,2] and Andrey Bogdanov[2]

[1] Shanghai Jiao Tong University, Department of Computer Science and Engineering, N0.800, Dongchuan Road, Shanghai, 200240, China.
[2] Katholieke Universiteit Leuven, ESAT/COSIC and IBBT, Kasteelpark Arenberg 10, 3001 Leuven, Belgium.
{qingju.wang, andrey.bogdanov}@esat.kuleuven.be

Abstract. CLEFIA is a popular recent block cipher designed by Sony Corporation, accepted as a lightweight encryption algorithm of the new ISO/IEC 29192-2 standard, and proposed as a Japanese e-Government recommendation cipher CRYPTREC candidate.
Provable security properties of cryptographic design are crucial in any security evaluation. Providing lower bounds on the number of active S-boxes in differential and linear characteristics has been one of the few important provable properties that can be formally shown for block ciphers and hence received a lot of attention.
In this work, we prove tighter lower bounds on the number of linearly active S-boxes in CLEFIA-type generalized Feistel networks (GFNs) with diffusion switching mechanism (DSM). We show that every 6 rounds of such GFNs provide 50% more linearly active S-boxes than proven previously. Moreover, we experimentally demonstrate that the new bound is tight for up to at least 12 rounds, whereas the previous one is not.
Thus, this paper delivers first *provable* evidence that diffusion switching mechanism actually provides an advantage by guaranteeing more active S-boxes in GFNs.

Keywords: block ciphers, generalized Feistel networks, CLEFIA, diffusion switching mechanism, substitution diffusion networks, linear cryptanalysis, efficiency

1 Introduction

1.1 Motivation

Generalized Feistel networks (GFNs) [18] have been popular with the designers of symmetric-key cryptographic primitives including block ciphers, stream ciphers and hash functions. They offer a simple way of domain extension given a function with good cryptographic properties. Probably the best understood structure of

its round transform relies on substitution-diffusion functions (SD-func-tions) — a brick layer of local nonlinear permutations (*S-boxes*) followed by a multiplication by a *diffusion matrix* over a binary finite field (*linear diffusion*).

GFN_4 are 4-line generalized Feistel networks. Type-I and type-II GFN_4 are referred to as GFN_4-I and GFN_4-II throughout this paper, structures are shown in Fig. 1 and 2. The findings of [5] indicate that going from single SD-functions [11, 19] to double SD-functions improves the efficiency of GFN_4 by up to 33% for GFN_4-I and by up to 50% for GFN_4-II, as measured by the proportion of differentially and linearly S-boxes in all S-boxes of the cipher. The work [5] proves that for GFN_4 with double SD-functions every 14 rounds of GFN_4-I and every 6 rounds of GFN_4-II add $7\mathcal{B}$ and $6\mathcal{B}$ differentially and linearly active S-boxes, respectively, where $\mathcal{B}$ is the branch number of the diffusion matrix M (or its transpose) used in the round functions. Underlying SD-type functions can differ

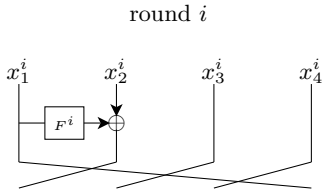


Fig. 1: GFN_4-I

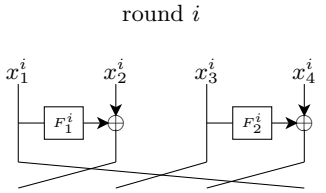


Fig. 2: CLEFIA-type GFN_4-II

depending on:

- **Number of distinct diffusion matrices:** The standard approach is to use a single matrix in all rounds and functions (*single-round diffusion*), e.g. applied in Camellia [1]. The alternative approach proposed in [13] is to employ two and more distinct diffusion matrices in different rounds and functions (*multiple-round diffusion*, or *diffusion switching mechanism, DSM*), which prevents difference and linear mask cancelation at XORs, e.g. utilized in CLEFIA [14].
- **Number of SD-layers in a function:** SD-type functions usually consist of a single SD-layer (*single SD-functions*), as e.g. those in CLEFIA [14] and Camellia [1]. In some ciphers, however, SD-type functions have double SD-layers (*double SD-functions*), e.g. in E2 [8] and Piccolo [12].

CLEFIA is a popular recent block cipher designed by Sony Corporation, accepted as a lightweight encryption algorithm of the new ISO/IEC 29192-2 standard, and proposed as a CRYPTREC Japaneese e-Government recommendation cipher. The design of CLEFIA is a 4-line type-II GFN (*GFN_4-II*) with DSM and single SD-functions. GFN_4-II belongs to the type of GFNs. We will be investigating in this paper – d-line type-II GFNs with DSM and single SD-functions, GFN_d-II.

1.2 Previous work on GFN$_d$-II

GFN$_d$-II with other types of SD-type functions have been thoroughly studied in the literature, see Table 1.

Tight lower bounds on the number of both differentially and linearly active S-boxes for GFN$_4$-II with single SD-functions and single-round diffusion are obtained in [11]: Every 6 rounds of GFN$_4$-II are proven to provide at least $2\mathcal{B}$ active S-boxes, where $\mathcal{B}$ is the differential and linear branch number of the diffusion matrices used in the round functions.

Tight minimum numbers of differentially and linearly active S-boxes for GFN$_4$-II with double SD-functions and single-round diffusion are proven in [5]. The findings of [5] indicate that going from single SD-functions [11] to double SD-functions improves the efficiency of GFN$_4$-II by up to 50%, as measured by the proportion of differentially and linearly S-boxes in all S-boxes of the cipher. The work [5] proves that every 6 rounds of GFN$_4$-II with double SD-functions add at least $6\mathcal{B}$ active S-boxes for both differential and linear cryptanalysis.

Bounds on the number of differentially and linearly active S-boxes for GFN$_4$-II with single SD-functions and DSM were obtained in [15]. It is proven that every 6 rounds add at least $2\mathcal{B}$ differentially and linearly active S-boxes. However, this bound is not tight, especially for the number of linearly active S-boxes. In fact, the bound proven in [15] for DSM yields a lower number of active S-boxes than for single-round diffusion. So there has been no proof so far that DSM has any advantage over single-round diffusion. This paper will greatly improve upon this.

Table 1: Upper bounds on the number of linearly active S-boxes and efficiency E (Definition 1) for GFN$_d$-II with SD-type functions.

design	rounds	bound	function	diffusion	tightness	efficiency E
GFN$_4$-II [11]	6	$2\mathcal{B}+2$	single SD	single-round	yes	1/6
GFN$_4$-II [5]	6	$6\mathcal{B}$	double SD	single-round	yes	1/4*
GFN$_d$-II [15]	6	$2\mathcal{B}$	single SD	DSM	no	1/6
GFN$_d$-II, here	6	$3\mathcal{B}$	single SD	DSM	yes	1/4

*Note that one has a doubled number of S-boxes in a round for double SD-functions
$\mathcal{B}$: See Definitions 3 and 4

1.3 Contributions of this paper

In this work, we prove that every 6 rounds of GFN$_d$-II with multiple-round diffusion (diffusion switching mechanism) and single SD-functions add at least $3\mathcal{B}$ linearly active S-boxes, see Table 1. This is exactly the construction behind the design of the lightweight block cipher CLEFIA, for the case of $d = 4$. $\mathcal{B}$ is the branch number of the single- and multiple-round diffusion matrices (their transposed inverses) used in the round functions. We experimentally demonstrate that the new bound is tight for up to at least 12 rounds, whereas the previous one is not.

The relevance of this bound is three-fold:

– This result indicates that the efficiency of CLEFIA-type GFNs is 50% higher than previously proven, in terms of the proportion of linearly active S-boxes in all S-boxes. This efficiency metric E is a valid efficiency metric introduced in [16] and used in [2–5]. Its definition can be found in Definition 1.
– Moreover, the new result suggests that the efficiency of GFN_d with SD-type functions is equally improved both by moving from single to double SD-functions [5] and by going from single-round diffusion to DSM over multiple rounds – the central contribution of this paper.
– The new bounds is also the first provable evidence that GFNs with DSM can actually provide more active S-boxes than GFNs with single-round diffusion. Previously [14, 15], for GFNs, this advantage has been only demonstrated numerically for some concrete CLEFIA-like examples.

2 Preliminaries

2.1 GFN_4 with SD-functions

Type-I and type-II GFNs are block ciphers with the state equally divided into an even number $d \geq 4$ wide lines. They are referred to as GFN_d-I and GFN_d-II in this paper. The structurs of GFN_d-I and GFN_d-II when $d = 4$ are as shown in Fig. 1 and 2. In one round of both GFN_4-I and GFN_4-II, let the input x^i of round i be $x^i = (x_1^i, x_2^i, x_3^i, x_4^i)$. Then the output of GFN_4-I and and for GFN_4-II will be $(x_2^i \oplus F^i(x_1^i), x_3^i, x_4^i, x_1^i)$ and $(x_2^i \oplus F_1^i(x_1^i), x_3^i, x_4^i \oplus F_2^i(x_3^i), x_1^i)$ respectively, for some keyed nonlinear functions F^i, F_1^i and F_2^i. The j-th F-functions F_j^i of

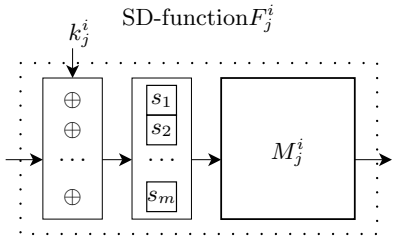


Fig. 3: SD-function

round i often exhibit the Substitution-Diffusion (SD) structure
(For type-I, there is only one F-function in each round). Here, the subkey addition followed by a layer of m S-boxes, $s_i, i = 1, \ldots, m$, and an $m \times m$ linear diffusion mapping M_j^i over a binary finite field. Such F-functions are called *SD-functions*. The structure of SD-functions is depicted in Fig. 3.

2.2 Diffusion Switching Mechanism

Diffusion Switching Mechanism (DSM) is a design approach for Feistel networks proposed by Shirai and Shibutani [13] and used in CLEFIA [14]. In this technique, two or more distinct diffusion matrices in the round function are switched

among multiple rounds in a predefined order to prevent the difference (linear mask) cancellation which occurs in the differential (linear) trails due to the fact that only a single matrix is employed for linear diffusion. Therefore, it provides a larger number of active S-boxes than the single-round diffusion and is adopted to enhance the efficiency of Feistel ciphers against differential and linear cryptanalysis. A linearly active S-box is defined as an S-box given the non-zero input linear mask.

2.3 Efficiency Metrics

Here a definition of two popular efficiency metrics for ciphers with SD-functions is given as introduced by Shirai and Preneel in [16]:

Definition 1. (Efficiency metrics [16]) *The efficiency metric $E_{m,r}$ for a GFN$_d$-II cipher over r rounds is defined as*

$$E_{m,r} = \frac{A_{m,r}}{S_{m,r}}$$

where $A_{m,r}$ is the number of active S-boxes over r rounds and $S_{m,r}$ is the total number of S-boxes over r rounds. The efficiency metric E is defined as

$$E = \lim_{m,r\to\infty} E_{m,r},$$

which simplifies comparisons.

3 Minimum number of active S-boxes

In this section, we prove a lower bound on the number of linearly active S-boxes for six rounds GFN$_d$-II ($d \geq 4$) with DSM, the untwisted form of which is illustrated in Fig. 4. When the DSM design strategy is applied to GFN$_d$-II, from Fig. 4 we can see that the relation between two matrices M_j^i in F_j^i and M_{j-1}^{i+2} in F_{j-1}^{i+2} for all possible i and j should be considered.

3.1 Branch numbers

First we recall the notions of branch numbers we will be using throughout the section:

Definition 2. (Bundle weight [6]) *Let $x \in \{0,1\}^{pn}$ be represented as $x = (x_0, x_1, \cdots, x_{p-1})$, where $x_i \in \{0,1\}^n$, then the bundle weight $w_n(x)$ is defined as*

$$w_n(x) = \#\{i | 0 \leq i \leq p-1, x_i \neq 0\}.$$

Definition 3. (Branch number [6]) *Let permutation $P : \{0,1\}^{pn} \to \{0,1\}^{qn}$. The branch number of P is defined as*

$$\mathcal{B}_n(P) = \min_{a\neq 0}\{w_n(a) + w_n(P(a))\}.$$

Similarly to Definition 3, one can define a linear branch number for two matrices:

Definition 4. (Linear branch number of multiple matrices [16]) *The linear branch number of two matrices is defined as*

$$\mathcal{B}^L = min\{\mathcal{B}_n[{}^t(M_j^i)^{-1}|{}^t(M_{j-1}^{i+2})^{-1}]\}.$$

where ${}^tM$ *denotes the transpose of matrix* M*,* $[M|N]$ *denotes an* $m \times 2m$ *matrix obtained by concatenating two* $m \times m$ *matrices* M *and* N*.*

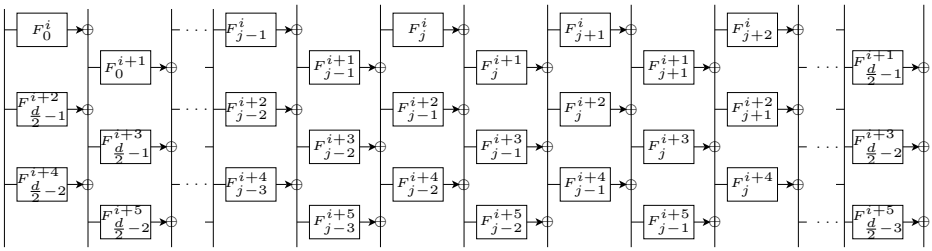


Fig. 4: 6-round GFN$_d$-II with DSM

3.2 Two basic properties

With respect to the j-th F-function F_j^i in the i-th round, we denote Γx_j^i and Γy_j^i as the input and output linear masks, L_j^i as the number of linearly active S-boxes in F_j^i. Also L^i denotes the number of linearly active S-boxes in the i-th round and, thus, $L^i = \sum_{j=0}^{\frac{d}{2}-1} L_j^i$. When DSM design strategy is employed in GFN$_d$-II, the following properties hold:

Property 1. Any two consecutive rounds of GFNs have at least one linear active F-function if a non-zero linear mask input is given.

If the input linear mask in the first round is non-zero, the input lines added to the output of F-funtion in the first round will become the input of the F-function in the next round, it means that the input of both F-funtions in the consecutive two rounds can not be zero at the same time.

The following property is derived in [15]:

Property 2. For any i and j, the set of L_j^i, L_j^{i+1} and L_{j-1}^{i+2} satisfies one of the two cases:

- $L_j^i = L_j^{i+1} = L_{j-1}^{i+2} = 0$;
- $L_j^i + L_j^{i+1} + L_{j-1}^{i+2} \geq \mathcal{B}^L$, where two of the terms are always non-zero.

Three round linear mask relation $\Gamma x_j^{i+1} = \Gamma y_j^i \oplus \Gamma y_{j-1}^{i+2}$ can be represented in matrix form as $\Gamma x_j^{i+1} = {}^t(M_j^i)^{-1}\Gamma x_j^i \oplus {}^t(M_{j-1}^{i+2})^{-1}\Gamma x_{j-1}^{i+2}$. If any two of the above terms are zero, it is obviously all the three are zero. Using the the notion of the branch number L_j^i, it implies the first case in Property 2; otherwise if any of the three terms is non-zero, we can see at least two terms will be non-zero. In addition, from

$$\Gamma x_j^{i+1} = [{}^t(M_j^i)^{-1} | {}^t(M_{j-1}^{i+2})^{-1}] \begin{pmatrix} \Gamma x_j^i \\ \Gamma x_{j-1}^{i+2} \end{pmatrix}$$

and since $[{}^t(M_j^i)^{-1} | {}^t(M_{j-1}^{i+2})^{-1}]$ has a branch number at least $\mathcal{B}^L$, we obtain

$$w_n(\Gamma x_j^i) + w_n(\Gamma x_j^{i+1}) + w_n(\Gamma x_{j-1}^{i+2}) \geq \mathcal{B}^L,$$

and, thus, the second case of Property 2 is shown.

3.3 Main result

Based on the above two properties, now we can derive the main result of the paper:

Theorem 1. *Let $d \geq 4$. Any consecutive six rounds of GFN$_d$-II with DSM guarantee at least $3\mathcal{B}^L$ linear active S-boxes.*

Proof. We consider six consecutive rounds starting in the i-th round. Property 1 implies that at least one F-function has a non-zero linear mask in the 3rd or 4th round, i.e. the $(i+2)$-th or the $(i+3)$-th round. We consider them separately as Case 1 and Case 2.

Case 1. A non-zero linear mask exists in the 3rd round, i.e. $L_j^{i+2} \neq 0$.
Based on condition $L_j^{i+2} \neq 0$ and Property 2, we get

$$L_{j+1}^i + L_{j+1}^{i+1} + L_j^{i+2} \geq \mathcal{B}^L \tag{1}$$

and $L_j^{i+1} + L_{j-1}^{i+3} \geq 1$. Now we consider two subcases $L_j^{i+1} \neq 0$ and $L_{j-1}^{i+3} \neq 0$ as follows:

Case 1-1. If $L_j^{i+1} \neq 0$, Property 2 implies that

$$L_j^i + L_j^{i+1} + L_{j-1}^{i+2} \geq \mathcal{B}^L. \tag{2}$$

Similarly, from condition $L_j^{i+2} \neq 0$ and Property 2, $L_{j-1}^{i+4} + L_j^{i+3} \geq 1$, i.e. at least one of the inequalities can be obtained from $L_{j-1}^{i+4} \neq 0$ and $L_j^{i+3} \neq 0$ respectively:

$$L_{j-1}^{i+3} + L_{j-1}^{i+4} + L_{j-2}^{i+5} \geq \mathcal{B}^L \tag{3}$$

$$L_j^{i+3} + L_j^{i+4} + L_{j-1}^{i+5} \geq \mathcal{B}^L \tag{4}$$

If $d = 4$, F_j^{i+4} will be F_{j-2}^{i+4}. Thus (4) will be $L_j^{i+3} + L_{j-2}^{i+4} + L_{j-1}^{i+5} \geq \mathcal{B}^L$. Therefore from the above inequalities (1), (2) and (3) or (1), (2) and (4)(even when $d = 4$, there is no overlapped term in the three inequalities), we get that in this case the minimum number of linearly active S-boxes is $\sum_{k=i}^{i+5} \sum_{j=0}^{\frac{d}{2}-1} L_j^k \geq 3\mathcal{B}^L$.

Case 1-2. If $L_{j-1}^{i+3} \neq 0$, Property 2 guarantees (3) again. From condition $L_{j-1}^{i+3} \neq 0$ and Property 2, $L_{j-1}^{i+2} + L_{j-2}^{i+4} \geq 1$ can be obtained. Therefore at least one of the two inequalities (2) and (5) with

$$L_{j-2}^{i+3} + L_{j-2}^{i+4} + L_{j-3}^{i+5} \geq \mathcal{B}^L \tag{5}$$

can be derived from Property 2, since of $L_{j-1}^{i+2} \neq 0$ and $L_{j-2}^{i+4} \neq 0$, respectively. If $d = 4$, F_{j-2}^{i+3} and F_{j-3}^{i+5} will become F_j^{i+3} and F_{j-1}^{i+5} respectively, and (5) will be $L_j^{i+3} + L_{j-2}^{i+4} + L_{j-1}^{i+5} \geq \mathcal{B}^L$. Therefore based on the above inequalities (1), (3) and (2) or inequalities (1), (3) and (5) (when $d = 4$, there are still no overlapped terms), we can conclude that in this case $\sum_{k=i}^{i+5} \sum_{j=0}^{\frac{d}{2}-1} L_j^k \geq 3\mathcal{B}^L$.

Case 2. Any non-zero linear mask exists in the 4th round, i.e. $L_{j-1}^{i+3} \neq 0$.
Since $L_{j-1}^{i+3} \neq 0$, from Property 2, (3) and $L_{j-1}^{i+2} + L_{j-2}^{i+4} \geq 1$ can be derived. Again, similarly to Case 1, two subcases will be considered as follows, $L_{j-1}^{i+2} \neq 0$ and $L_{j-2}^{i+4} \neq 0$:

Case 2-1. If $L_{j-1}^{i+2} \neq 0$, Property 2 implies (2). Similarly based on condition $L_j^{i+2} \neq 0$ and Property 2, at least one of the two inequalities (5) and (6) with

$$L_{j-1}^{i} + L_{j-1}^{i+1} + L_{j-2}^{i+2} \geq \mathcal{B}^L \tag{6}$$

can be obtained. When $d = 4$, (3) becomes $L_{j-1}^{i+3} + L_{j-3}^{i+4} + L_{j-2}^{i+5} \geq \mathcal{B}^L$. Thus the above inequalities (3), (2) and (5) or inequalities (3), (2) and (6), guarantee that the minimum number of active S-boxes is $\sum_{k=i}^{i+5} \sum_{j=0}^{\frac{d}{2}-1} L_j^k \geq 3\mathcal{B}^L$.

Case 2-2. If $L_{j-2}^{i+4} \neq 0$, Property 2 means that (5) holds. Moreover, from condition $L_{j-1}^{i+3} \neq 0$ and Property 2, $L_j^{i+1} + L_j^{i+2} \geq 1$ can be obtained. Therefore at least one of the two equalities (2) and (1) can be derived from Property 2 since $L_j^{i+1} \neq 0$ and $L_j^{i+2} \neq 0$, respectively. When considering $d = 4$, (5) will be turned into $L_j^{i+3} + L_{j-2}^{i+4} + L_{j-1}^{i+5} \geq \mathcal{B}^L$. Therefore inequalities (3), (5) and (1) or inequalities (3), (5) and (2), yield $\sum_{k=i}^{i+5} \sum_{j=0}^{\frac{d}{2}-1} L_j^k \geq 3\mathcal{B}^L$.

Combining all the above cases, we can conclude that at least $3\mathcal{B}^L$ linear active S-boxes are guaranteed in any six consecutive rounds of GFN$_d$-II with DSM ($d \geq 4$) which yields the claim of the theorem.

Therefore, we have proven that every six rounds GFN$_d$-II with DSM ($d \geq 4$) provide a tight upper bound of $3\mathcal{B}$ on the number of linearly active S-boxes. When $d = 4$, it is CLEFIA-type GFNs and, thus, we showed that any six rounds of CLEFIA-type GFNs yield by 50% more linearly active S-boxes than previously proven in [15], see also Table 1.

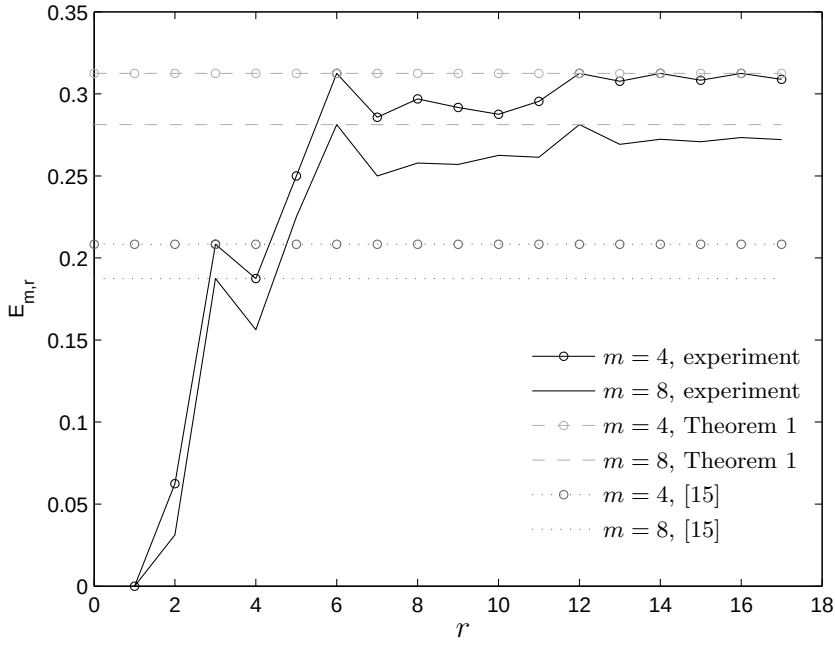


Fig. 5: Experimental efficiency $E_{m,r}$ and bounds, $m \in \{4, 8\}$, for GFN_4-II. The new bound is tight e.g. for $r = 6$ and $r = 12$

4 Tightness, experiments and conclusions

We investigate two examples of GFN_4-II with DSM and MDS diffusion matrices [6,9] to show tightness of the bounds of Theorem 1: one with $m = 4$ and one with $m = 8$. Note that $\mathcal{B}^L = m + 1$ for MDS matrices. For these constructions, we experimentally derive the actual numbers $A_{m,r}$ of linearly active S-boxes over several rounds r. Then, using metric $E_{m,r}$ of Definition 1, we numerically compute the efficiency of this construction over r rounds. These results are given in Fig. 5, together with the bounds of Theorem 1, $3\mathcal{B}^L = 3(m+1)$, as well as those of [15], $2\mathcal{B}^L = 2(m+1)$. To obtain the experimental results of Fig. 5, we adopt the mixed-integer linear programming (MILP) technique proposed in [10] to find the security bounds for the initialization phase of the Hitachi-designed stream cipher Enocoro-128v2 [17] against both differential and linear cryptanalysis. The optimizer CPLEX [7] is used in our implementation of the technique.

We observe that our bound appears tight for at least both $r = 6$ and $r = 12$ rounds, whereas the bound of [15] is not. Moreover, we see that considering multiples of 6 rounds for the bound on the number of linearly active S-boxes

is the most informative choice since it is those numbers of rounds that provide most efficiency.

We conclude with the observation that the bound of Theorem 1 delivers first provable evidence that employing the diffusion switching mechanism in GFN designs (and especially, in CLEFIA-type GFNs) actually provides an advantage by guaranteeing more active S-boxes.

Acknowledgements Andrey Bogdanov is postdoctoral fellow of the Fund for Scientific Research - Flanders (FWO). This work was supported in part by the National Natural Science Foundation of China (No. 61073150), by the IAP Programme P6/26 BCRYPT of the Belgian State, by the European Commission under contract number ICT- 2007-216676 ECRYPT NoE phase II, by KU Leuven-BOF (OT/08/027), and by the Research Council KU Leuven (GOA TENSE).

References

1. Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T.: Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms – Design and Analysis. In: D.R. Stinson and S. Tavares, editors, SAC'00, vol. 2012 of LNCS, pp. 39–56, Springer-Verlag (2001)
2. Bogdanov, A.: On the Differential and Linear Efficiency of Balanced Feistel Networks. Information Processing Letters 110(20), pp. 861–866, Elsevier (2010)
3. Bogdanov, A.: On Unbalanced Feistel Networks with Contracting MDS Diffusion. Designs, Codes and Cryptography 59(1-3), pp. 35–58. Springer-Verlag (2011)
4. Bogdanov, A., Shibutani, K.: Double SP-Functions: Enhanced Generalized Feistel Networks. In: U. Parampalli and P. Hawkes (eds.), ACISP 2011, LNCS, vol. 6812, pp. 106–119, Springer-Verlag (2011)
5. Bogdanov, A., Shibutani, K.: Generalized Feistel Networks Revisited. WCC'11, Workshop on Coding and Cryptography (2011). Submitted to Designs, Codes and Cryptography (2011)
6. Daemen, J., Rijmen, V.: The Design of Rijndael: AES — The Advanced Encryption Standard. Information Security and Cryptography, Springer-Verlag (2002)
7. IBM: IBM ILOG CPLEX Optimizer.
8. Kanda, M., Moriai, S., Aoki, K., Ueda, H., Takashima, Y., Ohta, K., Matsumoto, T.: E2 – A New 128-Bit Block Cipher. IEICE Trans. Fundamentals, vol. E83-A(1), January 2000.
9. Lidl, R., Niederreiter, H.: Finite Fields. Encyclopedia of Mathematics and Its Applications 20. Cambridge University Press (1997)
10. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and Linear Cryptanalysis using Mixed-Integer Linear Programming. In: Inscrypt'11. LNCS, Springer-Verlag (2011)
11. Shibutani, K.: On the Diffusion of Generalized Feistel Structures Regarding Differential and Linear Cryptanalysis. In: SAC'10. LNCS, Springer-Verlag (2010)
12. Shibutani, K. Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: Piccolo: An Ultra-Lightweight Blockcipher. In: B. Preneel and T. Takagi, editors, CHES'11, vol. 6917 of LNCS, Springer-Verlag (2011)
13. Shirai, T., Shibutani, K.: Improving immunity of feistel ciphers against differential cryptanalysis by using multiple mds matrices. In: FSE'04, LNCS, vol. 3017, pp. 260-278. Springer-Verlag (2004)

14. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-Bit Blockcipher CLEFIA (Extended Abstract). In: FSE'07. LNCS, vol. 4593, pp. 181–195. Springer-Verlag (2007)
15. Shirai, T., Araki, K.: On Generalized Feistel Structures Using the Diffusion Switching Mechanism. IEICE Transactions 91-A(8), pp. 2120–2129 (2008)
16. Shirai, T., Preneel, B.: On Feistel Ciphers Using Optimal Diffusion Mappings Across Multiple Rounds. In: ASIACRYPT'04. LNCS, vol. 3329, pp. 1–15. Springer-Verlag (2004)
17. Watanabe, D., Okamoto, K., Kaneko, T.: A Hardware-Oriented Light Weight Pseudo-Random Number Generator Enocoro-128v2. In: The Symposium on Cryptography and Information Security. pp. 3D1-3 (2010)
18. Zheng, Y., Matsumoto, T., Imai, H.: On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses. In: CRYPTO'89. LNCS, vol. 435, pp. 461–480. Springer-Verlag (1989)
19. Wu, W., Zhang, W., Lin, D.: Security on Generalized Feistel Scheme with SP Round Function. I. J. Network Security 3(3), 215–224 (2006)

Chapter 8

Related-Key Rectangle Cryptanalysis of Rijndael-160 and Rijndael-192

Publication Data

Q. Wang, Z. Liu, D. Toz, K. Varıcı, D. Gu: Related-Key Rectangle Cryptanalysis of Rijndael-160 and Rijndael-192. *IET Information Security*, 9(5): 266–276, 2015.

Contributions

Major contributor.

- Do the experiments and construct the distinguishers.
- Write the text from the very beginning to Section 4.
- Verify the attacks of Section 5 and 6.

Related-Key Rectangle Cryptanalysis of Rijndael-160 and Rijndael-192

Qingju Wang[1,4], Zhiqiang Liu[1,4], Deniz Toz[1,2], Kerem Varici[1,3], and Dawu Gu[4]

[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University
800 Dongchuan Road, Minhang District, Shanghai, 200240, China
[2] Clear2Pay, De Kleetlaan 6A, 1831 Diegem, Brussels, Belgium
[3] ICTEAM-Crypto Group, Universite catholique de Louvain, 1348 Louvain-la-Neuve, Belgium
[4] ESAT/COSIC, KU Leuven and iMinds
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
qingju.wang@esat.kuleuven.be, ilu_zq@163.com
deniz.toz@gmail.com,kerem.varici@uclouvain.be,dwgu@sjtu.edu.cn

Abstract. In this paper we present the first related-key rectangle cryptanalysis of Rijndael-160/160 and Rijndael-192/192. Our attack on Rijndael-160/160 covers eight rounds. The attack complexities are $2^{126.5}$ chosen plaintexts, $2^{129.28}$ 8-round Rijndael-160/160 encryptions and $2^{132.82}$ bytes. Our attack on Rijndael-192/192 covers ten rounds. It requires 2^{179} chosen plaintexts, $2^{181.09}$ 10-round Rijndael-192/192 encryptions and $2^{185.59}$ bytes memory. These are the currently best cryptanalytic results on Rijndael-160/160 and Rijndael-192/192 in terms of the number of attacked rounds. Furthermore, our results show that the slow diffusion in the key schedule of Rijndael makes it a target for this type of analysis.

Keywords: Rijndael, related-key attack, rectangle cryptanalysis

1 Introduction

Block ciphers are used widely in cryptography to ensure confidentiality and authenticity. They are needed both in software and in hardware. For example, they are used in electronic payments or for wireless security. For different demands, different algorithms are designed [1–3] and they have been standardised as well. Apart from this main functionality, block ciphers are also used as underlying primitives in the design of hash functions or pseudo-random number generators.

Rijndael [2] is a block cipher designed by Daemen and Rijmen and is a substitution-permutation network following the wide-trail strategy. Both the block length and the key length can be any multiple of 32 bits, with a minimum of 128 bits and a maximum of 256 bits, independently of each other with key size greater than or equal to block size. The 128-bit block variant of Rijndael has been chosen as the Advanced Encryption Standard (AES) [4]. This paper deals with non-AES Rijndael variants, that is, Rijndael-b/k where b indicates the block size and k indicates the key size in bits.

Without doubt AES is one of the most well-studied block ciphers: since its introduction 15 years ago there has been extensive analysis of AES. Some prominent examples include square attacks, impossible differential attacks, boomerang attacks, rectangle attacks and meet-in-the-middle attacks in both the single-key and related-key settings [5–18].

On the other hand, the variants of Rijndael with larger block sizes have got arguably less attention from the cryptographic community. Current analysis includes several multiset and integral attacks [19–22], as well as impossible differential cryptanalysis [23–25]. A summary of these attacks and their time and data complexities are given in Table 1. An important motivation for the study of large-block Rijndael is the deployment of Rijndael-like permutations in the design of hash functions, Whirlwind [26], SHAvite-3 [27], Whirlpool [28], ECHO [29], PHOTON [30] and SHA-3 finalist Grøstl [31] constituting some especially interesting instances.

The rectangle attack [32] introduced by Biham et al. is a special type of differential cryptanalysis. The main idea of the attack is to divide the cipher E into two sub-ciphers E_0 and E_1 such that $E = E_1 \circ E_0$. The attacker then constructs two relatively short differentials for E_0 and E_1 instead of finding a long differential for the block cipher E. After that, a rectangle distinguisher for E can be established by combining these two short differentials delicately. This technique is useful when we have short differentials with high probability instead of long ones with lower differential probabilities.

In the related-key model, the attacker can decrypt/encrypt not only under the master key K, but also under the keys $f_1(K), f_2(K) \dots f_m(K)$, which are called related-keys. The relations f_i are chosen by the attacker in advance. The aim of the attacker is to recover the master keys. The first related-key attacks consider simple mappings, for example, rotations [33] and bit flips [34]. Recent attack on AES [18] exploits the difference not between the master keys but between the subkeys. The extra control might make the attack harder to mount in practice. However, the designers still usually make great efforts to build ideal primitives which can be used without further cryptanalysis to the applications of modes of operation or protocols. Related-key attacks are applied to the attacks on the protocols that use ciphers as a building block [35]. Contrasting to the single-key attack, related-key attack recovers a secret parameter of the protocol. Therefore resistance to related-key attack becomes one of the important design aims for block ciphers, actually this was also stated as one of the design goals of the Rijndael.

Moreover it is also possible to combine rectangle-type attack with related-key attack to derive a more efficient cryptanalytic approach [36]. Actually, this type of combined approach has been applied to various block ciphers and some intriguing results have been achieved for AES [18, 37] and KASUMI [38, 39].

Contributions. In this paper we propose related-key rectangle attacks on reduced-round versions of Rijndael-160/160 and Rijndael-192/192. Our attacks use the idea of switch technique [40] and local collisions in [18]. We construct 6 and 8-round rectangle distinguishers of Rijndael-160/160 and Rijndael-192/192, based

on which we attack 8 and 10 rounds of these two ciphers respectively. To our knowledge, our results are the best ones in terms of the number of attacked rounds. The attacks on Rijndael-160/160 and Rijndael-192/192 are summarized in Table 1.

Table 1: Summary of Attacks on Rijndael-160/160 and Rijndael-192/192

Cipher	No. of Rd	Complexity Time (EN)	Complexity Data (CP)	Complexity Memory (Bytes)	Attack Type	Ref.
Rijndael-160/160	6	2^{135}	$2^{105.5}$	-	Imp. Diff.	[23]
	6	$2^{81.9}$	2^{147}	-	Imp. Diff.	[24]
	7	2^{144}	$2^{130.6}$	2^{128}	Multiset	[20]
	7	$2^{81.9}$	2^{147}	-	Imp. Diff.	[24]
	7	2^{108}	$2^{94.6}$	2^{92}	Integral	[41]
	7	$2^{98.6}$	$2^{98.6}$	-	Integral	[22]
	8	$2^{129.28}$	$2^{126.5\dagger}$	$2^{132.82}$	**RK Rectangle**	**§5**
Rijndael-192/192	6	2^{151}	$2^{121.5}$	2^{93}	Imp. Diff.	[23]
	6	$2^{113.8}$	$2^{93.2}$	2^{93}	Imp. Diff.	[24]
	7	2^{120}	$2^{128} - 2^{119}$	2^{61}	Partial Sum	[19]
	7	2^{144}	$2^{130.6}$	2^{128}	Multiset	[20]
	7	$2^{66.6}$	$2^{66.6}$	-	Integral	[22]
	7	$2^{174.5}$	$2^{28.5}$	-	Integral	[22]
	8	2^{188}	$2^{128} - 2^{119}$		Partial Sum	[19]
	8	$2^{177.4}$	2^{158}	2^{157}	Imp. Diff.	[24]
	8	$2^{81.4}$	2^{179}	2^{61}	Imp. Diff.	[24]
	8	$2^{174.5}$	$2^{68.5}$	-	Integral	[22]
	8	$2^{162.6}$	$2^{162.6}$	-	Integral	[22]
	9	$2^{174.5}$	$2^{164.5}$	-	Integral	[22]
	10	$2^{181.09}$	$2^{179\ddagger}$	$2^{185.59}$	**RK Rectangle**	**§6**

CP: Chosen Plaintext; EN: Encryptions
$^{\dagger}2^{128.5}$ ciphertexts under 4 related keys; $^{\ddagger}2^{181}$ ciphertexts under 4 related keys

Outline. This paper is organized as follows. In Section 2 we give a brief description of Rijndael and the notations used in our analysis. Section 3 introduces the rectangle attack briefly and shows our rectangle distinguishers. We demonstrate our attacks on Rijndael-160/160 and Rijndael-192/192 in Sections 5 and 6, respectively. Finally, Section 7 concludes this paper.

2 Description of Rijndael and Notations

In Rijndael, each data block (plaintext, ciphertext, subkey, or intermediate step) is represented by a $4 \times N_b$ `state matrix` of bytes, where N_b is the block size divided by 32. The state is then transformed by iterating a round function. The round function is composed of the following four operations:

- `SubBytes (SB)` : a non-linear byte substitution (8×8-bit S-box) that acts on every byte of the state.
- `ShiftRows (SR)`: a cyclic shift of bytes in a row that acts individually on each of the last three rows of the state. The shift offset C_i of row i depends on the block length N_b (See Fig. 1).
- `MixColumns (MC)`: a linear transformation (based on an $[8, 4, 5]$ MDS code over $GF(2^8)$) that acts independently on every column of the state
- `AddRoundKey (AK)`: the exclusive-or of the round key with the intermediate state.

The number of rounds for the cipher N_r varies with N_b and N_k (the key size divided by 32). Before the first round, there exists a whitening layer consisting of `AddRoundKey` only, and in the last round the `MixColumns` operation is omitted. We assume that this is also the case for the reduced round versions of Rijndael.

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)
128 ►		160 ►		196 ►		224 ►	256 ►

(a) Byte index

N_k \ N_b	4	5	6	7	8
4	10	11	12	13	14
5	11	11	12	13	14
6	12	12	12	13	14
7	13	13	13	13	14
8	14	14	14	14	14

(b) Number of rounds

N_b	C_0	C_1	C_2	C_3
4	0	1	2	3
5	0	1	2	3
6	0	1	2	3
7	0	1	2	4
8	0	1	3	4

(c) Shift offsets

Fig. 1: Byte Index of the State Matrix and the Shift Offsets for Each Block Length N_b

Key Scheduling: The key schedule derives (N_r+1) b-bit round keys RK_0, RK_1 $\ldots RK_{N_r}$ from the master key. It consists of a linear array of 4-byte words denoted by $W[i]$ for $0 \leq i \leq N_b \cdot (N_r+1)$. The first N_k words $W[0] \| W[1] \| \cdots \| W[N_k - 1]$ are directly initialized with the words of the master key, while the remaining key words, $W[i]$ for $i \in [N_k, N_k \cdot (N_r + 1) - 1]$ are generated by the following algorithm:

`if` $(i \bmod N_k = 0)$ `then` $W[i] = W[i-N_k] \oplus SB(W[i-1] \lll 8) \oplus Rcon[i/N_k]$
`else if` $(N_k > 6$ and $i \bmod N_k = 4)$ `then` $W[i] = W[i-N_k] \oplus SB(W[i-1])$

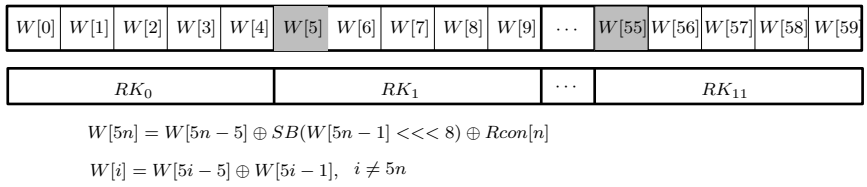


Fig. 2: Key expansion and round key selection for $N_b = N_k = 5$.

$$\texttt{else}\ W[i] = W[i - N_k] \oplus W[i - 1]$$

where $\lll$ denotes the rotation of the word to the left and $Rcon[\cdot]$ denotes the fixed constants. Then the round key RK_i is given by the words $W[N_b \cdot i]$ to $W[N_b \cdot (i + 1)]$. The key expansion and the the round key selection of Rijndael-160/160 are illustrated in Fig. 2. Here we only give a brief description of Rijndael, for more detailed specification of the cipher, we refer to [2].

Notation: The notation that we will use throughout this paper is as follows:

P_a, P_b, P_c, P_d	the plaintexts
$(P_a)_{i,j}$,$(P_b)_{i,j}$,$(P_c)_{i,j}$,$(P_d)_{i,j}$	the byte at row i column j of the plaintext state
C_a, C_b, C_c, C_d	the ciphertexts
K_a, K_b, K_c, K_d	secret related keys
$(K_a)_{i,j}$, $(K_b)_{i,j}$, $(K_c)_{i,j}$, $(K_d)_{i,j}$	the byte at row i column j of the secret related keys
K_a^r, K_b^r, K_c^r, K_d^r	secret subkey of K_a, K_b, K_c and K_d in round r
ΔK_{ab}^r, ΔK_{ac}^r, ΔK_{cd}^r, ΔK_{bd}^r	the difference of the related keys in round r
$(\Delta K_{ab}^r)_{i,j}$	difference byte of state ΔK_{ab}^r, at position row i and column j
$\texttt{SB}[(i,j)]$	the SB operation on the byte at row i column j of the state matrix
$\texttt{SB}[\{(i,j)\}]$	the SB operation on the subset bytes $\{(i,j)\}$ of the state matrix
$E(\cdot,\cdot)$	encryption operation defined as $\{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$

3 Rectangle Attack

The rectangle attack introduced by Biham et al. [32] aims to reduce the complexity of the differential cryptanalysis. The main idea is to use two short differential characteristics with high probabilities instead of one long characteristic with a lower probability. It is also possible to combine rectangle attack with related-key attack to derive the related-key rectangle attack [36] in which the attacker can query to the cipher with other keys that have a specified relation (often an xor-difference) with the original key.

Let the encryption function E of the block cipher be considered as a cascade of two sub-ciphers $E = E_1 \circ E_0$. Assume that there exists a related-key differential $\alpha \to \beta$ for E_0 under the key difference ΔK_{ab} with probability p, i.e., $(\Pr[E_0(P,K) \oplus E_0((P \oplus \alpha), (K \oplus \Delta K_{ab})) = \beta] = p)$. Similarly, assume that there exists a related-key differential $\gamma \to \delta$ for E_1 under the key difference ΔK_{ac} with probability q, where ΔK_{ab} and ΔK_{ac} are the key differences known by the attackers. A related-key rectangle distinguisher is then as follows:

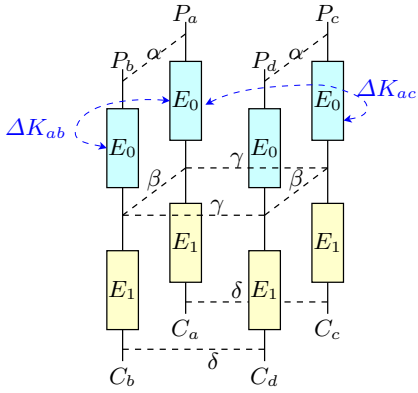


Fig. 3: The Related-Key Rectangle Distinguisher

1. Choose N plaintext pairs (P_a, P_b) with $P_b = P_a \oplus \alpha$ at random. Ask for the encryption of P_a under K_a and of P_b under K_b, respectively, where $K_b = K_a \oplus \Delta K_{ab}$.
2. Choose N plaintext pairs (P_c, P_d) with $P_d = P_c \oplus \alpha$ at random. Ask for the encryption of P_c under K_c and of P_d under K_d, respectively, where $K_c = K_a \oplus \Delta K_{ac}$ and $K_d = K_c \oplus \Delta K_{ab} = K_b \oplus \Delta K_{ac}$.
3. For a quartet of plaintexts (P_a, P_b, P_c, P_d) with corresponding ciphertexts (C_a, C_b, C_c, C_d), check whether $C_a \oplus C_c = C_b \oplus C_d = \delta$ holds or not. If yes, we call it a *right rectangle quartet*. Fig. 3 illustrates the related-key rectangle distinguisher.

The related-key rectangle attack can be mounted for all possible β's and γ's simultaneously. Firstly, we can use any γ for which $\gamma \xrightarrow{E_1} \delta$ holds. This is equivalent to mounting the attack for all values of γ with the condition $(E_0(P_a), E_0(P_c))$ and $(E_0(P_b), E_0(P_d))$ have the same difference (γ). In this case the probability that conditions of the distinguisher are satisfied is

$$2^{-n}p^2 \sum_{\gamma \xrightarrow{E_1} \delta} Pr^2[\gamma \xrightarrow{E_1} \delta] = 2^{-n}p^2\hat{q}^2,$$

where n is the block size, and $\hat{q} = \sqrt{\sum_{\gamma \xrightarrow{E_1} \delta} Pr^2[\gamma \xrightarrow{E_1} \delta]}$. Similarly, we can use all β values simultaneously as well. Conditions for this case become: $E_0(P_a) \oplus E_0(P_b) = E_0(P_c) \oplus E_0(P_d) = \beta$ and $E_0(P_a) \oplus E_0(P_c) = \gamma$ and the quartet has probability $Pr^2[\gamma \to \delta]$ to become a right quartet. Hence, the probability that a given quartet is a right quartet is

$$2^{-n} \sum_{\alpha \xrightarrow{E_0} \beta} Pr^2[\alpha \xrightarrow{E_0} \beta] \sum_{\gamma \xrightarrow{E_1} \delta} Pr^2[\gamma \xrightarrow{E_1} \delta] = 2^{-n}\hat{p}^2\hat{q}^2,$$

where $\hat{p} = \sqrt{\sum_{\alpha \xrightarrow{E_0} \beta} Pr^2[\alpha \xrightarrow{E_0} \beta]}$. Therefore starting with N plaintext pairs with difference α, we expect to find $N^2 2^{-n} (\hat{p}\hat{q})^2$ right quartets. For an ideal cipher, Step 3 is expected to hold with probability 2^{-2n}. Therefore, if $\hat{p}\hat{q} \gg 2^{-n/2}$, the algorithm above allows to distinguish E from an ideal cipher. We refer to [32, 36, 42, 43] for more detail.

3.1 Local Collision

A local collision is a differential that starts and ends with the zero difference in the internal state, but is non-zero in the middle. The idea of local collisions has been first introduced by Joux and Chabaud [44] to attack hash functions. It aims to inject a difference into an intermediate step and then to *correct* the resulting differences with the injections in the next steps to obtain a collision. The goal is to reduce the complexity of the attack by having as few disturbances as possible. This idea has been later successfully applied to block ciphers in [18]. A local collision of Rijndael-160/160 is shown in Fig. 4.

In order to construct an optimal trail, first of all we construct a minimal-weight disturbance layer, which will become a part of the key schedule difference. Then, correction layer is constructed by encrypting on the previous round of the disturbance layer. The key schedule difference is the sum of the disturbance and the correction layers. The 4-round Rijndael-160/160 key schedule difference constructed from local collisions is illustrated in Fig. 5. We follow this approach in our analysis of Rijndael-160/160 and Rijndael-192/192.

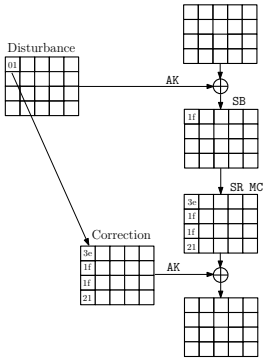


Fig. 4: A Local Collision of Rijndael-160/160

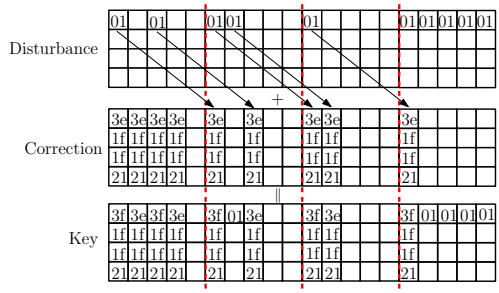


Fig. 5: Constructing Related-keys from Local Collisions

3.2 The Related Keys

In order to mount the related-key attacks presented in this paper, the adversary needs to construct the relations between different keys as follows.

Regarding Rijndael-160/160, for a secret key K_a, which the attacker tries to find, we define a simple form of this relation as xor with a constant to obtain K_b: $K_b = K_a \oplus \Delta K^0_{ab}$, where the constant ΔK^0_{ab} is chosen in advance (see Table 2). Then we compute the subkeys K^4_a and K^4_b, based on which, the subkeys K^4_c and K^4_d can be calculated by using the subkey difference ΔK^4_{ac}. After that, according to the key schedule of Rijndael-160/160, we can derive K_c and K_d from the subkeys K^4_c and K^4_d, respectively. This is depicted in Fig. 6.

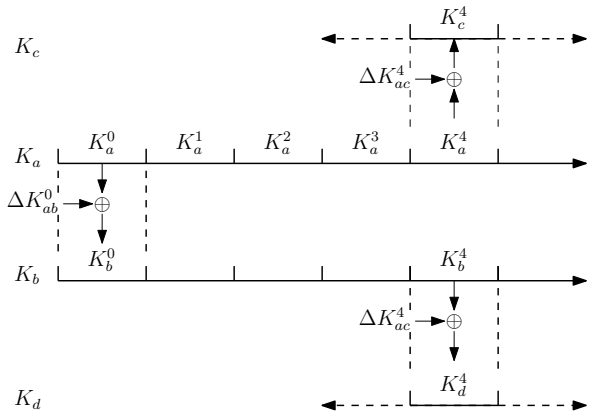


Fig. 6: The Related-Key Computation of Rijndael-160/160

For Rijndael-192/192, a more complex non-linear forms of the relation between the keys will be adopted. We choose a desired XOR relation in the second subkey as ΔK^1_{ab}, and then define the implied relation between the actual keys K_a and K_b as: $K_b = F^{-1}(F(K_a) \oplus \Delta K^1_{ab})$ where F represents a single round of the Rijndael-192/192 key schedule. Similar to Rijndael-160/160, we can define the relation between K_b, K_c and K_d for Rijndael-192/192 (see Table 3).

3.3 Rectangle Switch

In this Subsection we focus on the transition between the top characteristic E_0 and the bottom characteristic E_1 of the rectangle. This method is called *switch technique* [40] and it has been used to improve the probability of boomerang distinguisher [18, 39, 45]. In this paper, we apply this switch technique to our rectangle distinguishers on Rijndael. Let E be $(m + n)$-round Rijndael cipher. Then, the common application is to choose

$$E_0 = (AK \circ MC \circ SR \circ SB)^m$$
$$E_1 = (AK \circ MC \circ SR \circ SB)^n$$

However, due to the flexibility of the `SB` operation (i.e., it is applied to each byte in the state independently), this choice of E_0, E_1 can be done in a more

Table 2: Related-key Differences for Rijndael-160

ΔK^r_{ab}																
0	3f	3e	3f	3e	00	1	3f 01 3e 00 00				2	3f 3e 00 00 00			3	3f 01 01 01 01
	1f	1f	1f	1f	00		1f 00 1f 00 00					1f 1f 00 00 00				1f 00 00 00 00
	1f	1f	1f	1f	00		1f 00 1f 00 00					1f 1f 00 00 00				1f 00 00 00 00
	21	21	21	21	00		21 00 21 00 00					21 21 00 00 00				21 00 00 00 00
ΔK^r_{ac}																
4	21	21	21	21	00	5	21 00 21 00 00				6	21 21 00 00 00			7	21 00 00 00 00
	3e	3f	3e	3f	00		3e 01 3f 00 00					3e 3f 00 00 00				3e 01 01 01 01
	1f	1f	1f	1f	00		1f 00 1f 00 00					1f 1f 00 00 00				1f 00 00 00 00
	1f	1f	1f	1f	00		1f 00 1f 00 00					1f 1f 00 00 00				1f 00 00 00 00
8	$21 \oplus x^*$	$21 \oplus x$	$21 \oplus x$	$21 \oplus x$	$21 \oplus x$											
	3e	3f	3e	3f	3e											
	1f	1f	1f	1f	1f											
	1f	1f	1f	1f	1f											

* x might differ for ΔK^8_{ac} and ΔK^8_{bd}

Table 3: Related-key Differences for Rijndael-192

ΔK^r_{ab}																		
0	?	3e 00 00	3f 3e	1	3f	01	01	01	3e	00	2	3f 3e 3f 3e 00 00						
	?	1f 00 00	1f 1f		1f	00	00	00	1f	00		1f 1f 1f 1f 00 00						
	?	1f 00 00	1f 1f		1f	00	00	00	1f	00		1f 1f 1f 1f 00 00						
	?	21 00 00	21 21		21	00	00	00	21	00		21 21 21 21 00 00						
3	3f 01 3e 00 00 00			4	3f	3e	00	00	00	00	5	3f 01 01 01 01 01						
	1f 00 1f 00 00 00				1f	1f	00	00	00	00		1f 00 00 00 00 00						
	1f 00 1f 00 00 00				1f	1f	00	00	00	00		1f 00 00 00 00 00						
	21 00 21 00 00 00				21	21	00	00	00	00		21 00 00 00 00 00						
ΔK^r_{ac}																		
6	00 21 21 21 21 00			7	00	21	00	21	00	00	8	00 21 21 00 00 00						
	00 3e 3f 3e 3f 00				00	3e	01	3f	00	00		00 3e 3f 00 00 00						
	00 1f 1f 1f 1f 00				00	1f	00	1f	00	00		00 1f 1f 00 00 00						
	00 1f 1f 1f 1f 00				00	1f	00	1f	00	00		00 1f 1f 00 00 00						
9	00 21 00 00 00 00			10	x^*	$21 \oplus x$	$21 \oplus x$	$21 \oplus x$	$21 \oplus x$	$21 \oplus x$								
	00 3e 01 01 01 01				00	3e	3f	3e	3f	3e								
	00 1f 00 00 00 00				00	1f	1f	1f	1f	1f								
	00 1f 00 00 00 00				00	1f	1f	1f	1f	1f								

* x might differ for ΔK^{10}_{ac} and ΔK^{10}_{bd}

clever way. Let $\{(i,j)\}$ and $\{(i',j')\}$ be subsets of the state set, $\texttt{SB}[\{(i,j)\}]$ and $\texttt{SB}[\{(i',j')\}]$ be the SB operations on the bytes $\{(i,j)\}$ and $\{(i',j')\}$, respectively. Then E_0 and E_1 can alternatively be defined as follows:

$$E_0 = SB[\{(i,j)\}] \circ (AK \circ MC \circ SR \circ SB)^m,$$
$$E_1 = (AK \circ MC \circ SR \circ SB)^{n-1} \circ AK \circ MC \circ SR \circ SB[\{(i',j')\}], \qquad (1)$$

where $\{(i,j)\}$ is the absolute complement of $\{(i',j')\}$ in the state set, and the bytes (i,j) and (i',j') are passive in E_0 and E_1, respectively. For example, in our rectangle distinguisher of Rijndael-160/160, we take $m = 2, n = 4$. The first subcipher E_0 covers rounds 2–3 of Rijndael-160/160 and `SB` operations on 12 bytes (i,j) in round 4, where $1 \leq i \leq 3$ and $1 \leq j \leq 4$; The second subcipher E_1 starts with the `SB` operations on 8 bytes $(i',0)$, $(0,j')$ in round 4 ($1 \leq i' \leq 3$, $0 \leq j' \leq 4$), followed by $AK \circ MC \circ SR$ and rounds 5–7. Our 6-round rectangle distinguisher is illustrated in Fig. 7.

4 The Rectangle Distinguishers

In the analysis of Rijndael-160/160, we use a 6-round rectangle distinguisher, and extend one round before and after the distinguisher respectively (see Fig. 7). Our rectangle distinguisher covers rounds 2–7 and we use the switch technique in round 4 to avoid the active S-boxes in the key schedule and hence to reduce the complexity of our attack. We take $m = 2$ and $n = 4$ in Equation (1) to obtain E_0 and E_1.

The plaintext difference α is specified in 16 bytes (the left four columns), two of them(denoted as "?") can take any value whereas the remaining ones are fixed to `(0x3e,0x1f,0x1f,0x21)`$^{\mathtt{T}}$. The key difference is chosen such that when it is xored to the state, all differences cancel each other except the two bytes at $(0,0)$ and $(0,2)$ of the top characteristic. For the differences in these two active bytes we have:

$$(\mathtt{0x01} \oplus \alpha_{0,i}) \xrightarrow{SB} \mathtt{0x1f}, \qquad i \in \{0,2\} \tag{2}$$

This guarantees that the input differences to the S-box operations in all the internal states (except the ones specified in Equation 2) are `0x01`. For the active bytes in round 2 of the top characteristic, we adopt `0x1f` as the output difference of `SB` operation in order to achieve the optimal differential probability 2^{-6}. We develop the bottom characteristic by taking an similar approach. As to the active byte in round 3 of the top characteristic, there are 127 possibilities of the output difference for the input difference 0x01 according to the differential distribution table of `SB` operation, among which one happens with the probability 2^{-6}, the others happen with probability 2^{-7}. Then we construct the 6-round related-key rectangle distinguisher by combining the 127 top characteristics and one bottom characteristic mentioned above.

The probability of the 6-round distinguisher can be computed as follows.

- There are three active S-boxes in rounds 2–3, thus the probability of the 127 differentials for E_0 can be calculated as $\hat{p} = \sqrt{(2^{-6})^{2\cdot 2}[1 \cdot (2^{-6})^2 + 126 \cdot (2^{-7})^2]} \approx 2^{-10.5}$.
- There are five active S-boxes in rounds 4–7, thus the probability of the differential for E_1 is $\hat{q} = (2^{-6})^5 = 2^{-30}$.
- In total, the probability of this distinguisher can be calculated as $(2^{-10.5} \cdot 2^{-30})^2 \cdot 2^{-160} = 2^{-251}$.

Similarly, we find a 8-round rectangle distinguisher for Rijndael-192/192 which covers rounds 2–9, and the rectangle switch technique is applied at round 6 (see Fig. 8). There are 9 active S-boxes in the differential characteristics of E_0 (Note that for the active S-box in round 5, all the 127 possible output differences are used to derive 127 characteristics), and the probability can be computed as $\hat{p} = \sqrt{(2^{-6})^{2\cdot 8}[1 \cdot (2^{-6})^2 + 126 \cdot (2^{-7})^2]} \approx 2^{-51.5}$. For the differential characteristic of E_1, there are 5 active S-boxes and the probability is $\hat{q} = (2^{-6})^5 = 2^{-30}$. In total, the distinguisher holds with probability $(2^{-51.5} \cdot 2^{-30})^2 \cdot 2^{-192} = 2^{-355}$.

Moreover, the differences after the MC operations are given as:

$$\begin{pmatrix} \texttt{0x1f} \\ 0 \\ 0 \\ 0 \end{pmatrix} \xrightarrow{MC} \begin{pmatrix} \texttt{0x3e} \\ \texttt{0x1f} \\ \texttt{0x1f} \\ \texttt{0x21} \end{pmatrix}; \quad \begin{pmatrix} 0 \\ \texttt{0x1f} \\ 0 \\ 0 \end{pmatrix} \xrightarrow{MC} \begin{pmatrix} \texttt{0x21} \\ \texttt{0x3e} \\ \texttt{0x1f} \\ \texttt{0x1f} \end{pmatrix}$$

5 Attack on 8-Round Rijndael-160/160

By using the 6-round distinguisher for round 2–7 given in Subsection 4, we now present a key recovery attack on 8-round Rijndael-160/160 (round 1–8). Based on Fig. 7, an adversary aims to collect sufficient plaintext quartets such that among these plaintext quartets there are averagely 4 right quartets with respect to the 6-round distinguisher. Then among all the collected plaintext quartets, the expected number of quartets satisfying the input and output differences of this distinguisher for a random permutation is $(4/2^{-251}) \cdot 2^{-320} = 2^{-67}$. From this the adversary could distinguish the correct value from the wrong guessed values of the subkey bytes adopted in rounds 1 and 8 which are related to the 6-round distinguisher with a high probability. The attack procedure is divided into two phases: *data collection* phase and *key recovery* phase.

Data Collection

1. Collect N structures $G_i = \{U_i, V_i\}$ of 2^{17} plaintexts each, where $1 \le i \le N$, U_i, V_i are the sets of 2^{16} plaintexts of the form

?	c_1	?	c_2	c_3
c_4	c_5	c_6	c_7	c_8
c_9	c_{10}	c_{11}	c_{12}	c_{13}
c_{14}	c_{15}	c_{16}	c_{17}	c_{18}

and

?	$c_1 \oplus$ 3e	?	$c_2 \oplus$ 3e	c_3
$c_4 \oplus$ 1f	$c_5 \oplus$ 1f	$c_6 \oplus$ 1f	$c_7 \oplus$ 1f	c_8
$c_9 \oplus$ 1f	$c_{10} \oplus$ 1f	$c_{11} \oplus$ 1f	$c_{12} \oplus$ 1f	c_{13}
$c_{14} \oplus$ 21	$c_{15} \oplus$ 21	$c_{16} \oplus$ 21	$c_{17} \oplus$ 21	c_{18}

,

respectively, $c_i's$ $(1 \le i \le 18)$ are fixed 8-bit values and '?' takes all possible values.

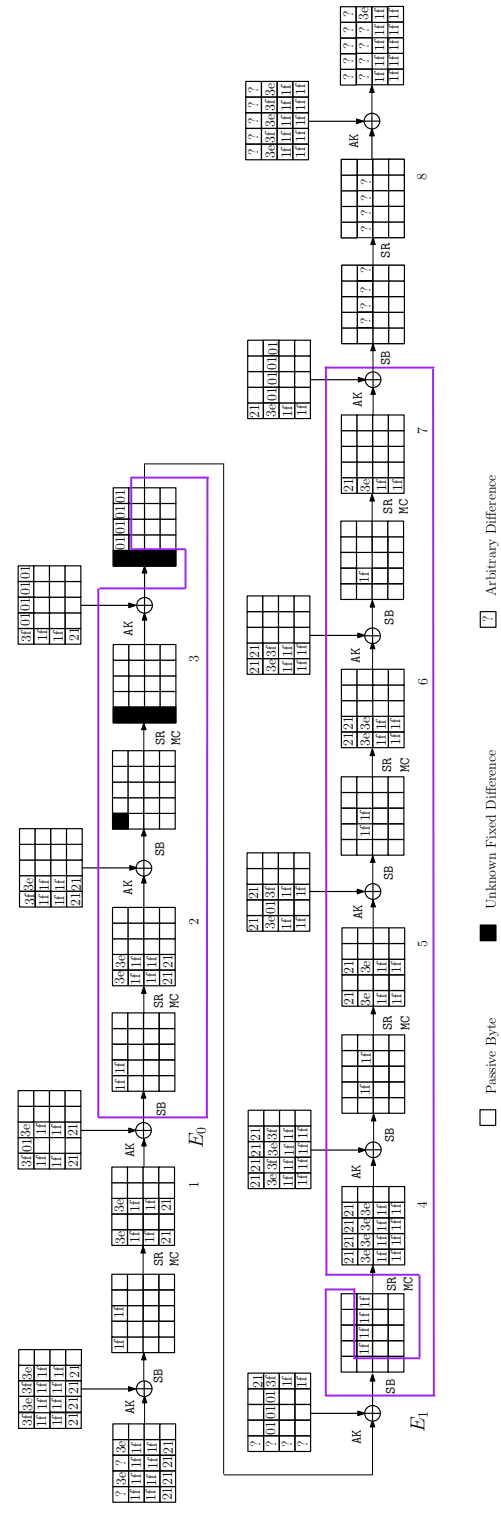


Fig. 7: The Related-key Rectangle Attack on 8-Round Rijndael-160/160. Switch is applied in Round 4

2. Let $T^{1a} = \{X_i^1\}_{1\le i\le N}$, $T^{1b} = \{Y_i^1\}_{1\le i\le N}$, $T^{2a} = \{X_i^2\}_{1\le i\le N}$, $T^{2b} = \{Y_i^2\}_{1\le i\le N}$, $T^{1c} = \{Z_i^1\}_{1\le i\le N}$, $T^{1d} = \{W_i^1\}_{1\le i\le N}$, $T^{2c} = \{Z_i^2\}_{1\le i\le N}$ and $T^{2d} = \{W_i^2\}_{1\le i\le N}$. Then for the case of $(T^{1a}, T^{1b}, T^{1c}, T^{1d})$, we can construct $(2^{16}\cdot 2^{16}\cdot N)^2 = 2^{64}\cdot N^2$ plaintext quartets meeting the input difference of round 1 given in Fig. 7. Similarly we can obtain $2^{64}\cdot N^2$ plaintext quartets satisfying the input difference of round 1 for $(T^{1a}, T^{1b}, T^{2c}, T^{2d})$, $(T^{2a}, T^{2b}, T^{1c}, T^{1d})$ and $(T^{2a}, T^{2b}, T^{2c}, T^{2d})$, respectively, resulting in $2^{64}\cdot N^2\cdot 4 = 2^{66}\cdot N^2$ plaintext quartets in total. Among these plaintext quartets there are about $2^{66}\cdot N^2\cdot(2^{-16})^2\cdot 2^{-251} = 2^{-217}\cdot N^2$ right quartets. Let $2^{-217}\cdot N^2 = 4$, we can deduce that $N = 2^{109.5}$.
3. Next, derive ciphertext quartets (C_a, C_b, C_c, C_d) and corresponding plaintext quartets (P_a, P_b, P_c, P_d) from $(T^{1a}, T^{1b}, T^{1c}, T^{1d})$, $(T^{1a}, T^{1b}, T^{2c}, T^{2d})$, $(T^{2a}, T^{2b}, T^{1c}, T^{1d})$ and $(T^{2a}, T^{2b}, T^{2c}, T^{2d})$ in an efficient way, such that (P_a, P_b, P_c, P_d), (C_a, C_b, C_c, C_d) satisfy the input and output differences of rounds 1 and 8, respectively, as shown in Fig. 7. Firstly, for the case of $(T^{1a}, T^{1b}, T^{1c}, T^{1d})$ do the following:
   - Initialize two vectors L^{ac} and L^{bd} consisting of 2^{88} lists L_η^{ac} and L_η^{bd}, respectively, where η corresponds to a 11-byte value (i.e., the bytes (1,4) and (i,j) of a ciphertext, where $2\le i\le 3$, $0\le j\le 4$), $L_\eta^{ac} = \{S_\eta^a, S_\eta^c, N_\eta^a, N_\eta^c\}$, $L_\eta^{bd} = \{S_\eta^b, S_\eta^d, N_\eta^b, N_\eta^d\}$, $S_\eta^a, S_\eta^b, S_\eta^c, S_\eta^d$ are the sets of ciphertexts under K_a, K_b, K_c and K_d, respectively, as well as their structure indices, and $N_\eta^a, N_\eta^b, N_\eta^c, N_\eta^d$ denote the cardinalities of the sets $S_\eta^a, S_\eta^b, S_\eta^c$ and S_η^d, respectively.
   - For each ciphertext in T^{1a}, extract the 88-bit value η, then insert the ciphertext and its structure index into the set S_η^a of the corresponding list L_η^{ac} and increase N_η^a by 1. For each ciphertext in T^{1c}, xor it with

00	00	00	00	00
00	00	00	00	3e
1f	1f	1f	1f	1f
1f	1f	1f	1f	1f

     and then extract the 88-bit value η. After that, insert the ciphertext and its structure index into the set S_η^c of the corresponding list L_η^{ac} and increase N_η^c by 1. Do similarly for the ciphertexts in T^{1b}, T^{1d} and update the lists L_η^{bd}.
   - Keep the lists L_η^{ac} in which both N_η^a and N_η^c are non-zero, and keep the lists L_η^{bd} in which both N_η^b and N_η^d are non-zero. Then derive the ciphertext quartets (C_a, C_b, C_c, C_d) from the remaining lists L_η^{ac} and L_η^{bd} by using following criteria:
     - C_a, C_c are chosen from the same list L_η^{ac}, and C_b, C_d come from the same list $L_{\eta'}^{bd}$.
     - The structure indices of C_a and C_b are the same, and the structure indices of C_c and C_d are the same.

     We obtain around $(2^{16}\cdot 2^{16}\cdot 2^{109.5})^2\cdot(2^{-88})^2 = 2^{107}$ ciphertext quartets (C_a, C_b, C_c, C_d) and their plaintext quartets (P_a, P_b, P_c, P_d) in this step.

– For each of the 2^{107} quartets (C_a, C_b, C_c, C_d), check whether the following conditions

$$(C_a \oplus C_c)_{0,0} = (C_a \oplus C_c)_{0,1} = \ldots = (C_a \oplus C_c)_{0,4}$$

and

$$(C_b \oplus C_d)_{0,0} = (C_b \oplus C_d)_{0,1} = \ldots = (C_b \oplus C_d)_{0,4}$$

hold or not. If not, discard the quartet. The expected number of remaining quartets after this step is about $2^{107} \cdot (2^{-32})^2 = 2^{43}$.

Do similarly for $(T^{1a}, T^{1b}, T^{2c}, T^{2d})$, $(T^{2a}, T^{2b}, T^{1c}, T^{1d})$ and $(T^{2a}, T^{2b}, T^{2c}, T^{2d})$, respectively, and finally we get about $2^{43} \cdot 4 = 2^{45}$ ciphertext quartets (C_a, C_b, C_c, C_d) and their plaintext quartets (P_a, P_b, P_c, P_d).

Key Recovery

5. Guess the subkey bytes $(K_a)_{0,j}$, $(K_c)_{0,j}$ $(j \in \{0, 2\})$ as follows:
   (a) guess $(K_a)_{0,0}$, $(K_c)_{0,0}$ and calculate the values of $(K_b)_{0,0}$, $(K_d)_{0,0}$ by using Table 2.
   (b) guess $(K_a)_{0,2}$, $(K_c)_{0,2}$ and derive the values of $(K_b)_{0,2}$, $(K_d)_{0,2}$ from Table 2.

   For each of the remaining quartets in substeps (a)–(b), test whether the corresponding equations

$$SB((P_a)_{0,j} \oplus (K_a)_{0,j}) \oplus SB((P_b)_{0,j} \oplus (K_b)_{0,j}) = \texttt{1f}$$
$$SB((P_c)_{0,j} \oplus (K_c)_{0,j}) \oplus SB((P_d)_{0,j} \oplus (K_d)_{0,j}) = \texttt{1f}$$

   are satisfied or not. If not, discard the quartet. After this step, the number of remaining quartets is about $2^{45} \cdot (2^{-8})^4 = 2^{13}$.

6. Guess the subkey bytes $(K_a^7)_{1,4}$, $(K_b^7)_{1,4}$, $(K_a^8)_{1,j}$, $(K_b^8)_{1,j}$ $(0 \le j \le 3)$ and obtain the values of $(K_c^7)_{1,4}$, $(K_d^7)_{1,4}$, $(K_c^8)_{1,j}$, $(K_d^8)_{1,j}$ according to Table 2. Then for each remaining quartet (C_a, C_b, C_c, C_d), verify whether the following equations

$$SB((K_a^7)_{1,4}) \oplus SB((K_c^7)_{1,4}) \oplus (C_a)_{0,0} \oplus (C_c)_{0,0} = \texttt{21}$$
$$SB((K_b^7)_{1,4}) \oplus SB((K_d^7)_{1,4}) \oplus (C_b)_{0,0} \oplus (C_d)_{0,0} = \texttt{21}$$
$$SB^{-1}((C_a)_{1,j} \oplus (K_a^8)_{1,j}) \oplus SB^{-1}((C_c)_{1,j} \oplus (K_c^8)_{1,j}) = \texttt{01}$$
$$SB^{-1}((C_b)_{1,j} \oplus (K_b^8)_{1,j}) \oplus SB^{-1}((C_d)_{1,j} \oplus (K_d^8)_{1,j}) = \texttt{01}$$

   hold or not. If not, remove the quartet.

7. If the number of the remaining quartets after above steps is two or more, output the corresponding 14 guessed subkey bytes $(K_a)_{0,j_1}$, $(K_c)_{0,j_1}$, $(K_a^7)_{1,4}$, $(K_b^7)_{1,4}$, $(K_a^8)_{1,j_2}$ and $(K_b^8)_{1,j_2}$ $(j_1 \in \{0, 2\},\ 0 \le j_2 \le 3)$ as the correct key information. Otherwise, return to Step 5 and repeat the procedure.

8. If the above 14 subkey bytes are retrieved after Step 7, perform an exhaustive search over all possible values of the remaining 128 bits of K_a^8 so as to recover the secret key.

5.1 Analysis of the attack

In Step 6, ten equations (each with probability 2^{-8}) need to be satisfied. Therefore, for a wrong guess of the above 14 subkey bytes, the expected number of quartets after Step 6 is $2^{13} \cdot (2^{-8})^{10} = 2^{-67}$. On the other hand, for a right guess of the key, the expected number of right quartets is about 4. This means that we can discard all the wrong subkeys (since the expected number of remaining quartets for a wrong subkey is 2^{-67}) and find the right 14 subkey bytes.

The probability of outputting a wrong key guess in Step 7 is derived by the following Poisson distribution:

$$X \sim Poi(\lambda = 2^{-67}).$$

As $\Pr[X \geq 2] \approx 2^{-135}$, the expected number of wrong key guesses suggested in Step 7 is about $(2^8)^{14} \cdot 2^{-135} = 2^{-23}$, and the wrong key information can be easily removed in Step 8. Similarly, the probability that two or more quartets remain after Step 7 for the correct key guess is also computed by the Poisson distribution:

$$X \sim Poi(\lambda = 4).$$

Since $\Pr[X \geq 2] \approx 0.91$, the success probability of the attack on 8-round Rijndael-160/160 is approximately 91%.

5.2 Complexity Issues

The data complexity of this attack is $2^{109.5} \cdot 2^{17} = 2^{126.5}$ chosen plaintexts which are encrypted under K_a, K_b, K_c and K_d, respectively (resulting in $2^{126.5} \cdot 4 = 2^{128.5}$ ciphertexts). The memory complexity is primarily owing to keeping T^{1a}, T^{1b}, T^{1c}, T^{1d}, T^{2a}, T^{2b}, T^{2c} and T^{2d}, thus it can be estimated as $8 \cdot 2^{125.5} \cdot 20 \approx 2^{132.82}$ bytes.

The time complexity of the attack can be derived as follows:

- For the *data collection* phase, the time complexity comes from Step 2 and Step 4.
  - The time complexity of Step 2 is $2^{126.5} \cdot 4 = 2^{128.5}$ 8-round Rijndael-160/160 encryptions.
  - The time complexity of Step 4 can be estimated as $2^{125.5} \cdot 8 = 2^{128.5}$ memory accesses, which can be measured as $2^{128.5} \cdot \frac{1}{20 \cdot 8} \approx 2^{121.18}$ 8-round Rijndael-160/160 encryptions.
- For the *key recovery* phase, the time complexity is calculated as follows:
  - The time complexity of Step 5 can be estimated as $2^{45} \cdot 2^{16} \cdot \frac{4}{20 \cdot 8} + 2^{29} \cdot 2^{32} \cdot \frac{4}{20 \cdot 8} \approx 2^{56.68}$ 8-round Rijndael-160/160 encryptions.
  - The time complexity of Step 6 can be estimated as $2^{13} \cdot 2^{112} \cdot \frac{20}{20 \cdot 8} = 2^{122}$ 8-round Rijndael-160/160 encryptions.
  - The time complexity of Step 8 is about 2^{128} 8-round Rijndael-160/160 encryptions.

As a result, the total time complexity of the attack is approximately $2^{129.28}$ 8-round Rijndael-160/160 encryptions.

6 Attack on 10-Round Rijndael-192/192

By using the 8-round distinguisher for round 2–9 given in Subsection 4, we now present a key recovery attack on 10-round Rijndael-192/192 (round 1–10). Based on Fig. 8, an adversary needs to collect sufficient plaintext quartets such that among these plaintext quartets there are averagely 8 right quartets with respect to the 8-round distinguisher. Then among all the collected plaintext quartets, the expected number of quartets satisfying the input and output differences of this distinguisher for a random permutation is $(8/2^{-355}) \cdot 2^{-384} = 2^{-26}$. From this the adversary could distinguish the correct value from the wrong guessed values of the subkey bytes adopted in rounds 1 and 10 which are related to the 8-round distinguisher with a high probability. The attack procedure is divided into two phases: *data collection* phase and *key recovery* phase.

Data Collection

1. Collect N structures $G_i = \{U_i, V_i\}$ of 2^{41} plaintexts each, where $1 \le i \le N$, U_i, V_i are the sets of 2^{40} plaintexts of the form

?	c_1	c_2	c_3	?	c_4
?	c_5	c_6	c_7	c_8	c_9
?	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}
?	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}

and

?	$c_1 \oplus$ 3e	c_2	c_3	?	$c_4 \oplus$ 3e
?	$c_5 \oplus$ 1f	c_6	c_7	$c_8 \oplus$ 1f	$c_9 \oplus$ 1f
?	$c_{10} \oplus$ 1f	c_{11}	c_{12}	$c_{13} \oplus$ 1f	$c_{14} \oplus$ 1f
?	$c_{15} \oplus$ 21	c_{16}	c_{17}	$c_{18} \oplus$ 21	$c_{19} \oplus$ 21

   respectively, $c_i's$ $(1 \le i \le 16)$ are fixed 8-bit values and '?' takes all possible values.
2. For each structure G_i
   (a) Ask for the encryption of U_i, V_i under K_a and K_b, respectively, to obtain $G_i^1 = \{X_i^1, Y_i^1\}$.
   (b) Ask for the encryption of V_i, U_i under K_a and K_b, respectively, to obtain $G_i^2 = \{X_i^2, Y_i^2\}$.
   (c) Ask for the encryption of U_i, V_i under K_c and K_d, respectively, to obtain $H_i^1 = \{Z_i^1, W_i^1\}$.
   (d) Ask for the encryption of V_i, U_i under K_c and K_d, respectively, to obtain $H_i^2 = \{Z_i^2, W_i^2\}$.
3. Let $T^{1a} = \{X_i^1\}_{1 \le i \le N}$, $T^{1b} = \{Y_i^1\}_{1 \le i \le N}$, $T^{2a} = \{X_i^2\}_{1 \le i \le N}$, $T^{2b} = \{Y_i^2\}_{1 \le i \le N}$, $T^{1c} = \{Z_i^1\}_{1 \le i \le N}$, $T^{1d} = \{W_i^1\}_{1 \le i \le N}$, $T^{2c} = \{Z_i^2\}_{1 \le i \le N}$ and $T^{2d} = \{W_i^2\}_{1 \le i \le N}$. Then for the case of $(T^{1a}, T^{1b}, T^{1c}, T^{1d})$, we can construct $(2^{40} \cdot 2^{40} \cdot N)^2 = 2^{160} \cdot N^2$ plaintext quartets meeting the input difference of round 1 given in Fig. 8. Similarly we can obtain $2^{160} \cdot N^2$ plaintext quartets satisfying the input difference of round 1 for $(T^{1a}, T^{1b}, T^{2c}, T^{2d})$, $(T^{2a}, T^{2b}, T^{1c}, T^{1d})$ and $(T^{2a}, T^{2b}, T^{2c}, T^{2d})$, respectively, resulting in $2^{160} \cdot N^2 \cdot 4 = 2^{162} \cdot N^2$ plaintext quartets in total. Among these plaintext quartets there are about $2^{162} \cdot N^2 \cdot (2^{-40})^2 \cdot 2^{-355} = 2^{-273} \cdot N^2$ right quartets. Let $2^{-273} \cdot N^2 = 8$, we can deduce that $N = 2^{138}$.

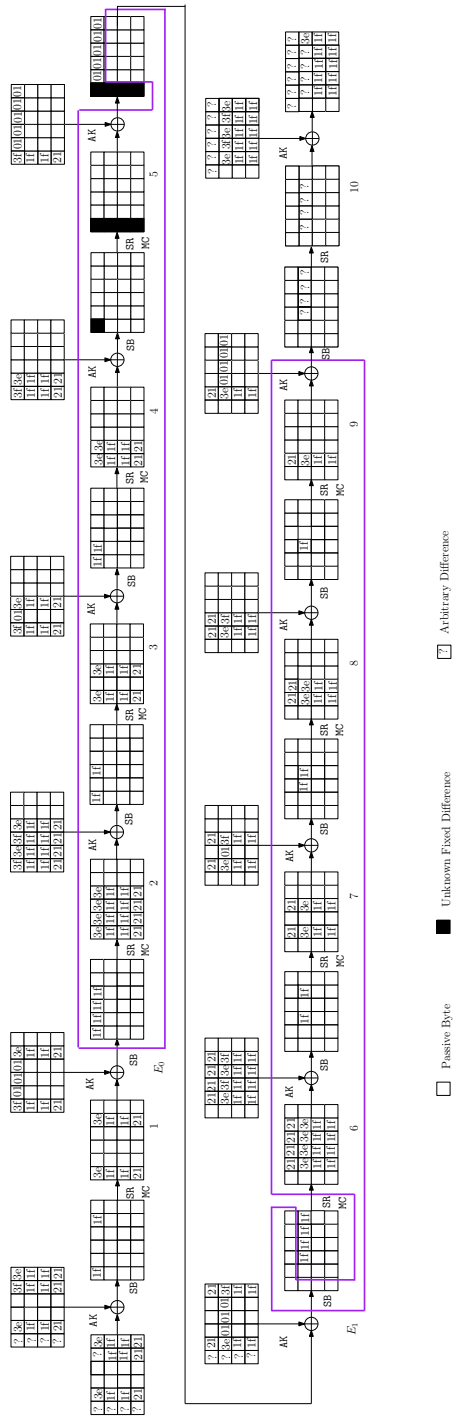


Fig. 8: The Related-key Rectangle Attack on 10-Round Rijndael-192/192. Switch is applied in Round 6

4. Next, derive ciphertext quartets (C_a, C_b, C_c, C_d) and corresponding plaintext quartets (P_a, P_b, P_c, P_d) from $(T^{1a}, T^{1b}, T^{1c}, T^{1d})$, $(T^{1a}, T^{1b}, T^{2c}, T^{2d})$, $(T^{2a}, T^{2b}, T^{1c}, T^{1d})$ and $(T^{2a}, T^{2b}, T^{2c}, T^{2d})$ in an efficient way, such that (P_a, P_b, P_c, P_d), (C_a, C_b, C_c, C_d) satisfy the input and output differences of rounds 1 and 10, respectively, as shown in Fig. 8. Firstly, for the case of $(T^{1a}, T^{1b}, T^{1c}, T^{1d})$ do the following:
   - Initialize two vectors L^{ac} and L^{bd} consisting of 2^{112} lists L^{ac}_{η} and L^{bd}_{η}, respectively, where η corresponds to a 14-byte value (i.e., the bytes (1,0), (1,5) and (i, j) of a ciphertext, where $2 \leq i \leq 3$, $0 \leq j \leq 5$), $L^{ac}_{\eta} = \{S^a_{\eta}, S^c_{\eta}, N^a_{\eta}, N^c_{\eta}\}$, $L^{bd}_{\eta} = \{S^b_{\eta}, S^d_{\eta}, N^b_{\eta}, N^d_{\eta}\}$, $S^a_{\eta}, S^b_{\eta}, S^c_{\eta}, S^d_{\eta}$ are the sets of ciphertexts under K_a, K_b, K_c and K_d, respectively, as well as their structure indices, and $N^a_{\eta}, N^b_{\eta}, N^c_{\eta}, N^d_{\eta}$ denote the cardinalities of the sets $S^a_{\eta}, S^b_{\eta}, S^c_{\eta}$ and S^d_{η}, respectively.

   - For each ciphertext in T^{1a}, extract the 112-bit value η, then insert the ciphertext and its structure index into the set S^a_{η} of the corresponding list L^{ac}_{η} and increase N^a_{η} by 1. For each ciphertext in T^{1c}, xor it with

00	00	00	00	00	00
00	00	00	00	00	3e
00	1f	1f	1f	1f	1f
00	1f	1f	1f	1f	1f

     and then extract the 112-bit value η. After that, insert the ciphertext and its structure index into the set S^c_{η} of the corresponding list L^{ac}_{η} and increase N^c_{η} by 1. Do similarly for the ciphertexts in T^{1b}, T^{1d} and update the lists L^{bd}_{η}.

   - Discard the lists L^{ac}_{η} in which N^a_{η} or N^c_{η} is 0, and remove the lists L^{bd}_{η} in which N^b_{η} or N^d_{η} is 0. For each remaining list L^{ac}_{η}, initialize 2^{48} lists $L^{ac}_{\eta,\theta} = \{S^a_{\eta,\theta}, S^c_{\eta,\theta}, N^a_{\eta,\theta}, N^c_{\eta,\theta}\}$ defined similarly to L^{ac}_{η}, where θ corresponds to a 6-byte value (i.e., the bytes $(0, j)$, $0 \leq j \leq 5$), then do the following:
     - For each ciphertext in S^a_{η}, extract the 48-bit value θ, then insert the ciphertext and its structure index into the set $S^a_{\eta,\theta}$ of the corresponding list $L^{ac}_{\eta,\theta}$ and increase $N^a_{\eta,\theta}$ by 1.
     - Let $\delta_1, \delta_2, \ldots, \delta_{127}$ denote all possible output differences of the S-Box for the input difference 01. For each ciphertext in S^c_{η}, xor it with

00	21	21	21	21	21
00	00	00	00	00	00
00	00	00	00	00	00
00	00	00	00	00	00

       and then extract the 48-bit value θ. After that, insert the ciphertext and its structure index into the set S^c_{η,θ_1}, S^c_{η,θ_2}, …, $S^c_{\eta,\theta_{127}}$ of the lists L^{ac}_{η,θ_1}, L^{ac}_{η,θ_2}, …, $L^{ac}_{\eta,\theta_{127}}$ and increase N^c_{η,θ_1}, N^c_{η,θ_2}, …, $N^c_{\eta,\theta_{127}}$ by 1, respectively, where $\theta_1, \ldots, \theta_{127}$ denote $\theta \oplus (\delta_1 \| \delta_1 \| \delta_1 \| \delta_1 \| \delta_1 \| \delta_1)$, …, $\theta \oplus (\delta_{127} \| \delta_{127} \| \delta_{127} \| \delta_{127} \| \delta_{127} \| \delta_{127})$, respectively.

     For each remaining list L^{bd}_{η}, do similarly to get the lists $L^{bd}_{\eta,\theta}$.

– Keep the lists $L^{ac}_{\eta,\theta}$ in which both $N^a_{\eta,\theta}$ and $N^c_{\eta,\theta}$ are non-zero, and keep the lists $L^{bd}_{\eta,\theta}$ in which both $N^b_{\eta,\theta}$ and $N^d_{\eta,\theta}$ are non-zero. Then derive the ciphertext quartets (C_a, C_b, C_c, C_d) from the remaining lists $L^{ac}_{\eta,\theta}$ and $L^{bd}_{\eta,\theta}$ by using following criteria:
  - C_a, C_c are chosen from the same list $L^{ac}_{\eta,\theta}$, and C_b, C_d come from the same list $L^{bd}_{\eta',\theta'}$.
  - The structure indices of C_a and C_b are the same, and the structure indices of C_c and C_d are the same.

With the above procedure we obtain around $(2^{40} \cdot 2^{40} \cdot 2^{138})^2 \cdot (2^{-112})^2 \cdot (\frac{127}{2^{48}})^2 \approx 2^{130}$ ciphertext quartets (C_a, C_b, C_c, C_d) and their plaintext quartets (P_a, P_b, P_c, P_d). Do similarly for $(T^{1a}, T^{1b}, T^{2c}, T^{2d})$, $(T^{2a}, T^{2b}, T^{1c}, T^{1d})$ and $(T^{2a}, T^{2b}, T^{2c}, T^{2d})$, respectively, and finally we get about $2^{130} \cdot 4 = 2^{132}$ ciphertext quartets (C_a, C_b, C_c, C_d) and their plaintext quartets (P_a, P_b, P_c, P_d).

Key Recovery

5. Guess the subkey bytes $(K_a)_{j,5}, (K_c)_{j,5}$ $(j \in \{2, 3, 0\})$ as follows:
   (a) Guess the subkey bytes $(K_a)_{2,5}$, $(K_c)_{2,5}$. According to the key schedule and Table 3, derive $(\Delta K_{ab})_{1,0}$, $(\Delta K_{cd})_{1,0}$ as below:

$$(\Delta K_{ab})_{1,0} = SB((K_a)_{2,5}) \oplus SB((K_a)_{2,5} \oplus 1\mathrm{f}) \oplus 1\mathrm{f},$$

$$(\Delta K_{cd})_{1,0} = SB((K_c)_{2,5}) \oplus SB((K_c)_{2,5} \oplus 1\mathrm{f}) \oplus 1\mathrm{f}.$$

   (b) Guess the subkey bytes $(K_a)_{3,5}$, $(K_c)_{3,5}$. Similarly, compute $(\Delta K_{ab})_{2,0}$ and $(\Delta K_{cd})_{2,0}$ in terms of the key schedule and Table 3.
   (c) Guess the subkey bytes $(K_a)_{0,5}$, $(K_c)_{0,5}$. Calculate $(\Delta K_{ab})_{3,0}, (\Delta K_{cd})_{3,0}$ from the key schedule and Table 3.

   For each of the remaining quartets in substeps (a)–(c), test whether the corresponding equations

$$(P_a)_{((j-1) \bmod 4),0} \oplus (P_b)_{((j-1) \bmod 4),0} \oplus (\Delta K_{ab})_{((j-1) \bmod 4),0} = 0$$
$$(P_c)_{((j-1) \bmod 4),0} \oplus (P_d)_{((j-1) \bmod 4),0} \oplus (\Delta K_{cd})_{((j-1) \bmod 4),0} = 0$$

   are satisfied or not. If not, discard the quartet. After this step, the number of remaining quartets is about $2^{132} \cdot (2^{-8})^6 = 2^{84}$.

6. Guess the subkey bytes $(K_a)_{0,4}$, $(K_c)_{0,4}$. Then for each remaining quartet (P_a, P_b, P_c, P_d), check whether the equations

$$SB((P_a)_{0,4} \oplus (K_a)_{0,4}) \oplus SB((P_b)_{0,4} \oplus (K_a)_{0,4} \oplus 3\mathrm{f}) = 1\mathrm{f}$$
$$SB((P_c)_{0,4} \oplus (K_c)_{0,4}) \oplus SB((P_d)_{0,4} \oplus (K_c)_{0,4} \oplus 3\mathrm{f}) = 1\mathrm{f}$$

   hold or not. If not, remove the quartet. The expected number of remaining quartets after this step is $2^{84} \cdot (2^{-8})^2 = 2^{68}$.

7. Guess the subkey bytes $(K_a)_{0,0}$, $(K_b)_{0,0}$, $(K_c)_{0,0}$, $(K_d)_{0,0}$. Then for each remaining quartet (P_a, P_b, P_c, P_d), test whether the equations
$$SB((P_a)_{0,0} \oplus (K_a)_{0,0}) \oplus SB((P_b)_{0,0} \oplus (K_b)_{0,0}) = 1\mathrm{f}$$
$$SB((P_c)_{0,0} \oplus (K_c)_{0,0}) \oplus SB((P_d)_{0,0} \oplus (K_d)_{0,0}) = 1\mathrm{f}$$
hold or not. If not, remove the quartet. The expected number of remaining quartets after this step is $2^{68} \cdot (2^{-8})^2 = 2^{52}$.

8. Guess the subkey bytes $(K_a^9)_{1,5}$, $(K_b^9)_{1,5}$. According to the key schedule and Table 3, derive $(\Delta K_{ac}^{10})_{0,0}$, $(\Delta K_{bd}^{10})_{0,0}$ as below:
$$(\Delta K_{ac}^{10})_{0,0} = SB((K_a^9)_{1,5}) \oplus SB((K_a^9)_{1,5} \oplus 01),$$
$$(\Delta K_{bd}^{10})_{0,0} = SB((K_b^9)_{1,5}) \oplus SB((K_b^9)_{1,5} \oplus 01).$$
Then for each remaining quartet (C_a, C_b, C_c, C_d), verify whether the equations
$$(C_a)_{0,0} \oplus (C_c)_{0,0} \oplus (\Delta K_{ac}^{10})_{0,0} = 0$$
$$(C_b)_{0,0} \oplus (C_d)_{0,0} \oplus (\Delta K_{bd}^{10})_{0,0} = 0$$
hold or not. If not, remove the quartet. Note that $(\Delta K_{ac}^{10})_{0,0}$, $(\Delta K_{bd}^{10})_{0,0} \in \{\delta_1, \ldots, \delta_{127}\}$, and from Step 4 we know that $(C_a)_{0,0} \oplus (C_c)_{0,0}$, $(C_b)_{0,0} \oplus (C_d)_{0,0} \in \{\delta_1, \ldots, \delta_{127}\}$, thus the expected number of remaining quartets after this step is $2^{52} \cdot (2^{-7})^2 = 2^{38}$. Moreover, according to Step 4 and Table 3 we have that for the remaining quartets, $(C_a)_{0,i} \oplus (C_c)_{0,i} = (\Delta K_{ac}^{10})_{0,i}$ and $(C_b)_{0,i} \oplus (C_d)_{0,i} = (\Delta K_{bd}^{10})_{0,i}$ hold for $1 \le i \le 5$.

9. Guess the subkey bytes $(K_a^{10})_{1,j}$, $(K_b^{10})_{1,j}$ $(1 \le j \le 4)$ as follows:
   (a) guess $(K_a^{10})_{1,1}$, $(K_b^{10})_{1,1}$ and calculate the values of $(K_c^{10})_{1,1}$, $(K_d^{10})_{1,1}$ by using Table 3.
   (b) guess $(K_a^{10})_{1,2}$, $(K_b^{10})_{1,2}$ and derive the values of $(K_c^{10})_{1,2}$, $(K_d^{10})_{1,2}$ from Table 3.
   (c) guess $(K_a^{10})_{1,3}$, $(K_b^{10})_{1,3}$, $(K_a^{10})_{1,4}$, $(K_b^{10})_{1,4}$ and use Table 3 to obtain the values of $(K_c^{10})_{1,3}$, $(K_d^{10})_{1,3}$, $(K_c^{10})_{1,4}$, $(K_d^{10})_{1,4}$.
   For each of the remaining quartets in substeps (a)–(c), check whether the corresponding equations
$$SB^{-1}((C_a)_{1,j} \oplus (K_a^{10})_{1,j}) \oplus SB^{-1}((C_c)_{1,j} \oplus (K_c^{10})_{1,j}) = 01$$
$$SB^{-1}((C_b)_{1,j} \oplus (K_b^{10})_{1,j}) \oplus SB^{-1}((C_d)_{1,j} \oplus (K_d^{10})_{1,j}) = 01$$
   are satisfied or not. If not, discard the quartet.

10. If the number of the remaining quartets after above steps is six or more, output the corresponding 22 guessed subkey bytes $(K_a)_{i,5}$, $(K_c)_{i,5}$, $(K_a)_{0,4}$, $(K_c)_{0,4}$, $(K_a)_{0,0}$, $(K_b)_{0,0}$, $(K_c)_{0,0}$, $(K_d)_{0,0}$, $(K_a^9)_{1,5}$, $(K_b^9)_{1,5}$, $(K_a^{10})_{1,j}$ and $(K_b^{10})_{1,j}$ $(i \in \{0, 2, 3\},\ 1 \le j \le 4)$ as the correct key information. Otherwise, return to Step 5 and repeat the procedure.

11. If the above 22 subkey bytes are retrieved after Step 10, perform an exhaustive search over all possible values of the remaining 152 bits of K_a so as to recover the secret key.

6.1 Analysis of the attack

The expected number of remaining quartets after each substep in Step 9 is given as (a) $2^{38} \cdot (2^{-8})^2 = 2^{22}$, (b) $2^{22} \cdot (2^{-8})^2 = 2^6$ and (c) $2^6 \cdot (2^{-8})^4 = 2^{-26}$, respectively. Thus for a wrong guess of the above 22 subkey bytes, the expected number of quartets after Step 9 is 2^{-26}. On the other hand, for a right guess of the key, the expected number of right quartets is 8.

The probability of outputting a wrong key guess in Step 10 is derived by the following Poisson distribution:

$$X \sim Poi(\lambda = 2^{-26}).$$

As $\Pr[X \geq 6] \approx 2^{-165.49}$, the expected number of wrong key guesses suggested in Step 10 is about $(2^8)^{22} \cdot 2^{-165.49} = 2^{10.51}$, and the wrong key information can be removed in Step 11. Similarly, the probability that six or more quartets remain after Step 9 for the correct key guess is also computed by the Poisson distribution:

$$X \sim Poi(\lambda = 8).$$

Since $\Pr[X \geq 6] \approx 0.81$, the success probability of the attack on 10-round Rijndael-192/192is approximately 81%.

6.2 Complexity Issues

The data complexity of this attack is $2^{138} \cdot 2^{41} = 2^{179}$ chosen plaintexts which are encrypted under K_a, K_b, K_c and K_d, respectively (resulting in $2^{179} \cdot 4 = 2^{181}$ ciphertexts). The memory complexity is primarily owing to keeping T^{1a}, T^{1b}, T^{1c}, T^{1d}, T^{2a}, T^{2b}, T^{2c} and T^{2d}, thus it can be estimated as $8 \cdot 2^{178} \cdot 24 \approx 2^{185.59}$ bytes.

The time complexity of the attack can be derived as follows:

- For the *data collection* phase, the time complexity comes from Step 2 and Step 4.
  - The time complexity of Step 2 is $2^{179} \cdot 4 = 2^{181}$ 10-round Rijndael-192/192encryptions.
  - The time complexity of Step 4 can be estimated as $2^{178} \cdot 8 = 2^{181}$ memory accesses, which can be measured as $2^{181} \cdot \frac{1}{24 \cdot 10} \approx 2^{173.09}$ 10-round Rijndael-192/192encryptions.
- For the *key recovery* phase, the time complexity is calculated as follows:
  - The time complexity of Step 5 can be estimated as $2^{132} \cdot 2^{16} \cdot \frac{4}{24 \cdot 10} + 2^{116} \cdot 2^{32} \cdot \frac{4}{24 \cdot 10} + 2^{100} \cdot 2^{48} \cdot \frac{4}{24 \cdot 10} \approx 2^{143.68}$ 10-round Rijndael-192/192encryptions.
  - The time complexity of Step 6 can be estimated as $2^{84} \cdot 2^{64} \cdot \frac{4}{24 \cdot 10} = 2^{142.09}$ 10-round Rijndael-192/192encryptions.
  - The time complexity of Step 7 can be estimated as $2^{68} \cdot 2^{96} \cdot \frac{4}{24 \cdot 10} = 2^{158.09}$ 10-round Rijndael-192/192encryptions.
  - The time complexity of Step 8 can be estimated as $2^{52} \cdot 2^{112} \cdot \frac{4}{24 \cdot 10} = 2^{158.09}$ 10-round Rijndael-192/192encryptions.

- The time complexity of Step 9 can be estimated as $2^{38} \cdot 2^{128} \cdot \frac{4}{24 \cdot 10} + 2^{22} \cdot 2^{144} \cdot \frac{4}{24 \cdot 10} + 2^{6} \cdot 2^{176} \cdot \frac{8}{24 \cdot 10} \approx 2^{177.09}$ 10-round Rijndael-192/192encryptions.
- The time complexity of Step 11 is about $2^{152} \cdot 2^{10.51} = 2^{162.51}$ 10-round Rijndael-192/192encryptions.

As a result, the total time complexity of the attack is approximately $2^{181.09}$ 10-round Rijndael-192/192encryptions.

7 Conclusion

In this paper, we performed key recovery attacks on reduced-round versions of Rijndael-160/160/160 and Rijndael-192/192. Firstly, we constructed a 6-round related-key rectangle distinguisher of Rijndael-160/160/160, with which we attacked 8 rounds of the cipher. Moreover, we established a 8-round related-key rectangle distinguisher of Rijndael-192/192, based on which we demonstrated the attack on 10 rounds of the cipher. Our results show that the related-key rectangle attack is one of the best methods to analyze Rijndael and Rijndael-like structures. To sum up, none of our attacks directly threatens the security of Rijndael but they reduce the security margin of the cipher.

Acknowledgement

The authors are grateful to all anonymous reviewers for their valuable comments. Moreover, the authors are supported by the National Natural Science Foundation of China (no. 61202371, 61402288), Major State Basic Research Development Program (973 Plan, no. 2013CB338004), China Postdoctoral Science Foundation (no. 2012M521829), Shanghai Postdoctoral Research Funding Program (no. 12R21414500), Plan of action for the innovation of science and technology of Shanghai Municipal Science and Technology Commission (no. 14511100300), and Shanghai Engineering Research Center Project (no. GCZX14014, C14001).

References

1. National Bureau of Standards. Data Encryption Standard. FIPS-Pub.46. National Bureau of Standards, U.S. Department of Commerce, Washington D.C., January 1977.
2. Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.
3. 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 *CHES*, pages 450–466, 2007.
4. FIPS 197. Advanced Encryption Standard. Federal Information Processing Standards Publication 197, U.S. Department of Commerce/N.I.S.T , 2001.
5. Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The Block Cipher Square. In Eli Biham, editor, *FSE*, volume 1267 of *Lecture Notes in Computer Science*, pages 149–165. Springer, 1997.

6. Henri Gilbert and Marine Minier. A Collision Attack on 7 Rounds of Rijndael. In *AES Candidate Conference*, pages 230–241, 2000.
7. Stefan Lucks. Attacking Seven Rounds of Rijndael under 192-bit and 256-bit Keys. In *AES Candidate Conference*, pages 215–229, 2000.
8. Alex Biryukov. The Boomerang Attack on 5 and 6-Round Reduced AES. In Hans Dobbertin, Vincent Rijmen, and Aleksandra Sowa, editors, *AES Conference*, volume 3373 of *Lecture Notes in Computer Science*, pages 11–15. Springer, 2004.
9. Raphael Chung-Wei Phan. Impossible Differential Cryptanalysis of 7-round Advanced Encryption Standard (AES). *Inf. Process. Lett.*, 91(1):33–38, 2004.
10. Wentao Zhang, Wenling Wu, and Dengguo Feng. New Results on Impossible Differential Cryptanalysis of Reduced AES. In Kil-Hyun Nam and Gwangsoo Rhee, editors, *ICISC*, volume 4817 of *Lecture Notes in Computer Science*, pages 239–250. Springer, 2007.
11. Hüseyin Demirci and Ali Aydin Selçuk. A Meet-in-the-Middle Attack on 8-Round AES. In Kaisa Nyberg, editor, *FSE*, volume 5086 of *Lecture Notes in Computer Science*, pages 116–126. Springer, 2008.
12. Jiqiang Lu, Orr Dunkelman, Nathan Keller, and Jongsung Kim. New Impossible Differential Attacks on AES. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *INDOCRYPT*, volume 5365 of *Lecture Notes in Computer Science*, pages 279–293. Springer, 2008.
13. Eli Biham, Orr Dunkelman, and Nathan Keller. Related-Key Impossible Differential Attacks on 8-Round AES-192. In David Pointcheval, editor, *CT-RSA*, volume 3860 of *Lecture Notes in Computer Science*, pages 21–33. Springer, 2006.
14. Wentao Zhang, Wenling Wu, Lei Zhang, and Dengguo Feng. Improved Related-Key Impossible Differential Attacks on Reduced-Round AES-192. In Eli Biham and Amr M. Youssef, editors, *Selected Areas in Cryptography*, volume 4356 of *Lecture Notes in Computer Science*, pages 15–27. Springer, 2006.
15. Jongsung Kim, Seokhie Hong, and Bart Preneel. Related-Key Rectangle Attacks on Reduced AES-192 and AES-256. In Alex Biryukov, editor, *FSE*, volume 4593 of *Lecture Notes in Computer Science*, pages 225–241. Springer, 2007.
16. Wentao Zhang, Lei Zhang, Wenling Wu, and Dengguo Feng. Related-Key Differential-Linear Attacks on Reduced AES-192. In K. Srinathan, C. Pandu Rangan, and Moti Yung, editors, *INDOCRYPT*, volume 4859 of *Lecture Notes in Computer Science*, pages 73–85. Springer, 2007.
17. Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolic. Distinguisher and Related-Key Attack on the Full AES-256. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 231–249. Springer, 2009.
18. Alex Biryukov and Dmitry Khovratovich. Related-Key Cryptanalysis of the Full AES-192 and AES-256. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2009.
19. Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David Wagner, and Doug Whiting. Improved Cryptanalysis of Rijndael. In Schneier [46], pages 213–230.
20. Jorge Nakahara, Daniel Santana de Freitas, and Raphael Chung-Wei Phan. New Multiset Attacks on Rijndael with Large Blocks. In Ed Dawson and Serge Vaudenay, editors, *Mycrypt*, volume 3715 of *Lecture Notes in Computer Science*, pages 277–295. Springer, 2005.
21. Samuel Galice and Marine Minier. Improving Integral Attacks Against Rijndael-256 Up to 9 Rounds. In Serge Vaudenay, editor, *AFRICACRYPT*, volume 5023 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2008.

22. Yanjun Li and Wenling Wu. Improved Integral Attacks on Rijndael. *Journal of Information Science and Engineering*, 27(6):2031–2045, 2011.
23. Jorge Nakahara and Ivan Carlos Pavão. Impossible-Differential Attacks on Large-Block Rijndael. In Juan A. Garay, Arjen K. Lenstra, Masahiro Mambo, and René Peralta, editors, *ISC*, volume 4779 of *Lecture Notes in Computer Science*, pages 104–117. Springer, 2007.
24. Lei Zhang, Wenling Wu, Je Hong Park, Bonwook Koo, and Yongjin Yeom. Improved Impossible Differential Attacks on Large-Block Rijndael. In Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen, and Der-Tsai Lee, editors, *ISC*, volume 5222 of *Lecture Notes in Computer Science*, pages 298–315. Springer, 2008.
25. Qingju Wang, Dawu Gu, Vincent Rijmen, Ya Liu, Jiazhe Chen, and Andrey Bogdanov. Improved Impossible Differential Attacks on Large-Block Rijndael. In Taekyoung Kwon, Mun-Kyu Lee, and Daesung Kwon, editors, *ICISC*, volume 7839 of *Lecture Notes in Computer Science*, pages 126–140. Springer, 2012.
26. 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.
27. Eli Biham and Orr Dunkelman. The SHAvite-3 Hash Function. Submission to NIST (Round 2), 2009.
28. P.S.L.M. Barreto and V. Rijmen. The whirlpool hashing function. Submitted to NESSIE (September 2000), 2000. `http://www.larc.usp.br/~pbarreto/WhirlpoolPage.html`(2008/12/11) (revised May 2003).
29. R. Benadjila, O. Billet, H. Gilbert, G. Macario-Rat, T. Peyrin, M. Robshaw, and Y. Seurin. SHA-3 proposal: ECHO. Submission to NIST (updated), 2009. `http://crypto.rd.francetelecom.com/echo/doc/echo_description_1-5.pdf`.
30. Jian Guo, Thomas Peyrin, and Axel Poschmann. The PHOTON Family of Lightweight Hash Functions. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 222–239. Springer, 2011.
31. Praveen Gauravaram, Lars R. Knudsen, K. Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. Grøstl - a SHA-3 candidate. Submission to NIST, 2008. `http://www.groestl.info`.
32. 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.
33. Eli Biham. New types of cryptanalytic attacks using related keys. *J. Cryptology*, 7(4):229–246, 1994.
34. John Kelsey, Bruce Schneier, and David Wagner. Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA. In Yongfei Han, Tatsuaki Okamoto, and Sihan Qing, editors, *Information and Communication Security, First International Conference, ICICS'97, Beijing, China, November 11-14, 1997, Proceedings*, volume 1334 of *Lecture Notes in Computer Science*, pages 233–246. Springer, 1997.
35. Erik Tews, Ralf-Philipp Weinmann, and Andrei Pyshkin. Breaking 104 bit WEP in less than 60 seconds. In Sehun Kim, Moti Yung, and Hyung-Woo Lee, editors, *Information Security Applications, 8th International Workshop, WISA 2007, Jeju Island, Korea, August 27-29, 2007, Revised Selected Papers*, volume 4867 of *Lecture Notes in Computer Science*, pages 188–202. Springer, 2007.

36. Eli Biham, Orr Dunkelman, and Nathan Keller. Related-Key Boomerang and Rectangle Attacks. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 507–525. Springer, 2005.
37. Alex Biryukov and Ivica Nikolic. Automatic Search for Related-Key Differential Characteristics in Byte-Oriented Block Ciphers: Application to AES, Camellia, Khazad and Others. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 322–344. Springer, 2010.
38. Eli Biham, Orr Dunkelman, and Nathan Keller. A Related-Key Rectangle Attack on the Full KASUMI. In Bimal K. Roy, editor, *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2005.
39. Orr Dunkelman, Nathan Keller, and Adi Shamir. A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 393–410. Springer, 2010.
40. David Wagner. The Boomerang Attack. In Lars R. Knudsen, editor, *FSE*, volume 1636 of *LNCS*, pages 156–170. Springer, 1999.
41. Marine Minier and Benjamin Pousse. Improving Integral Cryptanalysis against Rijndael with Large Blocks. *CoRR*, abs/0910.2153, 2009.
42. John Kelsey, Tadayoshi Kohno, and Bruce Schneier. Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent. In Schneier [46], pages 75–93.
43. Eli Biham, Orr Dunkelman, and Nathan Keller. New Combined Attacks on Block Ciphers. In Henri Gilbert and Helena Handschuh, editors, *FSE*, volume 3557 of *LNCS*, pages 126–144. Springer, 2005.
44. Florent Chabaud and Antoine Joux. Differential Collisions in SHA-0. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 56–71. Springer, 1998.
45. Alex Biryukov, Christophe De Cannière, and Gustaf Dellkrantz. Cryptanalysis of SAFER++. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 195–211. Springer, 2003.
46. Bruce Schneier, editor. *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, volume 1978 of *Lecture Notes in Computer Science*. Springer, 2001.

Chapter 9

PRIMATEs

Publication Data

E. Andreeva, B. Bilgin, A. Bogdanov, A. Luykx, F. Mendel, B. Mennink, N. Mouha, Q. Wang, K. Yasuda: PRIMATEs v1.02. Round 2 of CAESAR Competition, 2015.

Contributions

Major contributor of the PRIMATEs permutation:

- Design the operation ShiftRows, do the experiments and write the text.
- Work on security analysis of PRIMATE and write the text, together with Florian Mendel.

PRIMATEs v1.02

Submission to the CAESAR Competition

Designers/Submitters:

Elena Andreeva[1], Begül Bilgin[1,2], Andrey Bogdanov[3], Atul Luykx[1], Florian Mendel[4], Bart Mennink[1], Nicky Mouha[1], Qingju Wang[1,5], and Kan Yasuda[6]

[1] KU Leuven, ESAT/COSIC, and iMinds Belgium.
[2] University of Twente, EEMCS/SCS, The Netherlands.
[3] Technical University of Denmark, DTU Compute, Denmark.
[4] Graz University of Technology, IAIK, Austria.
[5] Shanghai Jiao Tong University, Department of Computer Science and Engineering, China.
[6] NTT Secure Platform Laboratories, Japan.

primates.ae
primates@esat.kuleuven.be

Changes From v1.01

1. Clarification of the bounds on collision producing trails (Sect. 4.4.2).

Changes From v1.0

1. Included a ranking of the parameters as required by the CAESAR competition.
2. Removed statements describing APE's nonce as being optional.
3. Improved the descriptions of the algorithms, layout, and wording.
4. Added figures for both fractional and integral, associated data and message cases.

All security analysis performed on the schemes in v1.0 and the reference software implementation provided as v1 also holds for the schemes in v1.01 and v1.02.

Notation

Set $\mathbf{K} := \{0,1\}^k$, $\mathbf{T} := \{0,1\}^\tau$, $\mathbf{N} := \{0,1\}^\nu$, $\mathbf{R} := \{0,1\}^r$, $\mathbf{C} := \{0,1\}^c$, and $\mathbf{C}^{\frac{1}{2}} := \{0,1\}^{c/2}$. Given the state $X \in \mathbf{R} \times \mathbf{C}$, $X_r \in \mathbf{R}$ denotes its rate part and $X_c \in \mathbf{C}$ its capacity part. We write $0^r \in \mathbf{R}$ for a shorthand of $00\cdots0 \in \mathbf{R}$.

The bitwise XOR operation of the bit strings a_1 and a_2 is denoted by $a_1 \oplus a_2$. Both $a_1 \parallel a_2$ and $a_1 a_2$ denote the concatenation of the bit strings a_1 and a_2.

An element of $\mathbf{R}$ is called a block. Let $\mathbf{R}^*$ denote the set of strings whose length is a non-negative multiple of r and at most $2^{c/2}$ blocks. Given a plaintext (message) $M \in \{0,1\}^*$, we divide it into blocks and write $M[1]M[2]\cdots M[w] \leftarrow M$, where each $M[i]$ for $i < w$ is a block and $M[w]$ is a string of length less than or equal to a block. We refer messages with $0 < |M[w]| < r$ as fractional messages (as opposed to integral messages where $|M[w]| = r$). When we write $M \parallel 10^*$, we mean that M is padded with a 1-bit and then zeros until the length of the resulting string is a multiple of r. By definition, an empty input message $M = \varnothing$ is also padded to the one zero-length block $M[1] \leftarrow 10^{r-1}$.

By $\lfloor M \rfloor_n$ we denote the n most significant bits of M (the n leftmost bits) whereas by $\lceil M \rceil_n$ we denote the n least significant bits.

Authenticated encryption with associated data (AEAD).

An authenticated encryption algorithm with associated data consists of a key generation $\mathcal{K}$, an encryption $\mathcal{E}$ and a decryption $\mathcal{D}$ algorithm. The encryption algorithm $\mathcal{E}$ takes as input a key $K \in \mathbf{K}$, associated data $A \in \mathbf{R}^*$, and a message $M \in \mathbf{R}^*$, and returns a ciphertext $C \in \mathbf{R}^*$ and a tag $T \in \mathbf{T}$, as $(C,T) \leftarrow \mathcal{E}_K(A,M)$. The decryption algorithm $\mathcal{D}$ takes as input a key $K \in \mathbf{K}$, associated data $A \in \mathbf{R}^*$, a ciphertext $C \in \mathbf{R}^*$, and a tag $T \in \mathbf{T}$, and returns either a message $M \in \mathbf{R}^*$ or the reject symbol $\perp$, as $M/\perp \leftarrow \mathcal{D}_K(A,C,T)$. The two functionalities $\mathcal{E}$ and $\mathcal{D}$ are sound, in the sense that

$$\mathcal{D}_K(A, \mathcal{E}_K(A,M)) = M,$$

for all K, A and M.

Nonce-based AEAD.

Whenever the AEAD scheme takes an additional nonce $N \in \mathbf{N}$ argument both in encryption and decryption we speak of nonce-based AEAD. The encryption algorithm is then defined as $(C,T) \leftarrow \mathcal{E}_K(N,A,M)$ and the decryption algorithm as $M/\perp \leftarrow \mathcal{D}_K(N,A,C,T)$ with the soundness condition $\mathcal{D}_K(N,A,\mathcal{E}_K(N,A,M)) = M$ holding for all N, K, A, and M.

Nonces.

A nonce $N \in \mathbf{N}$ is an unique non-repeating value, e.g. a counter. The nonces in this work are public values and we alternatively refer to them as public message numbers. We do not use secret message numbers. How the sender and receiver generate and synchronize nonces is left implicit as long as the uniqueness condition is satisfied.

	$s = 80$ bits	$s = 120$ bits
b (state size)	200 bits	280 bits
c (capacity size)	160 bits	240 bits
r (rate size)	40 bits	40 bits
permutations	PRIMATE-80	PRIMATE-120

Table 1: The security levels for the PRIMATEs family.

1 Parameters

The authenticated encryption family PRIMATEs is defined by the following two parameters:

1. The mode of operation $Scheme \in \{\mathsf{APE}, \mathsf{HANUMAN}, \mathsf{GIBBON}\}$.
2. The security level $s \in \{80, 120\}$ bits;

The security level will regularly be expressed in terms of bits (80 or 120 bits). The security level determines: the state size b, where the state consists of a rate part with r and a capacity part with c bits; and the permutation family PRIMATE-s. The PRIMATE-s : $\{0,1\}^b \rightarrow \{0,1\}^b$ family consists of four permutations p_1, p_2, p_3, and p_4. On the other hand, each mode of operation determines the key length k, the tag length τ, the nonce length ν, and the subset of permutations from PRIMATE-s.

For the purpose of the CAESAR competition, we rank the PRIMATEs as follows:

1. APE-120
2. HANUMAN-120
3. GIBBON-120
4. APE-80
5. HANUMAN-80
6. GIBBON-80

We note, however, that the different PRIMATEs serve different security goals, as we will clarify in this document.

1.1 Recommended Parameters

We recommend a security level s of either 80 or 120 bits for PRIMATEs family, as shown in Table 1.

In Table 2, we provide the respective key, tag and nonce values for the three modes where *Scheme*-s indicates the mode under $s = 80$ and $s = 120$ bits respectively. For HANUMAN and GIBBON we have identical values as compared to APE.

	APE-s	HANUMAN-s	GIBBON-s
k (key size)	$2s$	s	s
τ (tag size)	$2s$	s	s
ν (nonce size)	s	s	s
PRIMATE	p_1	p_1, p_4	p_1, p_2, p_3

Table 2: Key,tag and nonce values for the three modes of the PRIMATEs family.

When any of the parameters of the PRIMATEs are modified, a new key must be chosen uniformly at random. The length of plaintext and associated data processing is discussed in Sect. 2. Our recommendation for lightweight authenticated encryption is HANUMAN. For lightweight applications where speed is critical we recommend GIBBON and for lightweight environments where additional security requirements are needed or security is critical we recommend APE. The primary recommended security level is $s = 120$, whereas we recommend $s = 80$ for extremely lightweight applications. The primary recommended mode on the other hand is APE followed by HANUMAN and GIBBON. Hence, in this document, we prioritize security. APE is robust even when the nonce is misused whereas HANUMAN is secure as long as the nonce is unique and non-repeating. GIBBON is the most efficient at the cost of weaker security guarantees. As such, for different applications HANUMAN or GIBBON may be preferred over APE. We also note that a possible weakness observed in one of the modes, does not necessarily apply to the other modes.

2 Specification of PRIMATEs

We now provide the specifications of APE, HANUMAN and GIBBON. In this document, we specify that the message and associated data can consist of a non-integer number of bytes. However, due to restrictions on the API imposed by the CAESAR competition, all implementations that will be provided during the competition only support an integer number of bytes.

2.1 APE

The APE algorithm is described in Fig. 1. In APE, as opposed to HANUMAN and GIBBON, we treat the nonce the same way as the associated data whenever present. APE supports variable length associated data and plaintexts. As discussed in Sect. 3 we recommend the associated data and the plaintexts to be of size at most $2^{c/2}$ bits. For APE-80 this is approximately 2^{77} bytes and for APE-120 this is 2^{117} bytes. The APE algorithm uses the permutation p_1 together with its inverse p_1^{-1} for decryption. The key is used twice for: (1) part of the capacity of the initial state; and (2) after the tag generation. The fractional message cases are dealt with differently as compared to the integral data as elaborated below:

Consider a message M and denote its last block by $M[w]$, where $|M[w]| = |M| \bmod r$. We distinguish among three cases:

Algorithm 1: $\mathcal{E}_K(N, A, M)$

Input: $K \in \mathbf{C}$, $N \in \mathbf{C}^{\frac{1}{2}}$, $A \in \{0,1\}^*$, $M \in \{0,1\}^*$
Output: $C \in \{0,1\}^*$, $T \in \mathbf{C}$

$V \leftarrow 0^r \parallel K$
$N[1]N[2]\cdots N[y] \leftarrow N$
for $i = 1$ **to** y **do**
  $V \leftarrow p_1(N[i] \oplus V_r \parallel V_c)$
end
if $A \neq \varnothing$ **then**
  $A[1]A[2]\cdots A[u] \leftarrow A$
  $A[u] \leftarrow A[u] \parallel 10^*$
  for $i = 1$ **to** u **do**
    $V \leftarrow p_1(A[i] \oplus V_r \parallel V_c)$
  end
end
$V \leftarrow V \oplus (0^{b-1} \parallel 1)$
$M[1]M[2]\cdots M[w] \leftarrow M$
$l \leftarrow |M[w]|$
$M[w] \leftarrow M[w] \parallel 10^*$
for $i = 1$ **to** w **do**
  $V \leftarrow p_1(M[i] \oplus V_r \parallel V_c)$
  $C[i] \leftarrow V_r$
end
$C \leftarrow C[1]C[2]\cdots C[w-2]$
$C \leftarrow C \parallel \lfloor C[w-1] \rfloor_l$
$C \leftarrow C \parallel C[w]$
$T \leftarrow V_c \oplus K$
return (C, T)

Algorithm 2: $\mathcal{D}_K(N, A, C, T)$

Input: $K \in \mathbf{C}$, $N \in \mathbf{C}^{\frac{1}{2}}$, $A \in \{0,1\}^*$, $C \in \{0,1\}^*$, $T \in \mathbf{C}$
Output: $M \in \{0,1\}^*$ or $\perp$

$IV \leftarrow 0^r \parallel K$
$N[1]N[2]\cdots N[y] \leftarrow N$
for $i = 1$ **to** y **do**
  $IV \leftarrow p_1(N[i] \oplus IV_r \parallel IV_c)$
end
if $A = \varnothing$ **then**
  $A[1]A[2]\cdots A[u] \leftarrow A$
  $A[u] \leftarrow A[u] \parallel 10^*$
  for $i = 1$ **to** u **do**
    $IV \leftarrow p_1(A[i] \oplus IV_r \parallel IV_c)$
  end
end
$C[1]C[2]\cdots C[w] \leftarrow C$
$l \leftarrow |C[w]|$
$C[w] \leftarrow \lceil C[w-1] \rceil_{r-l} \parallel C[w]$
$C[w-1] \leftarrow \lfloor C[w-1] \rfloor_l$
$C[0] \leftarrow IV_r$
$V \leftarrow p_1^{-1}(C[w] \parallel K \oplus T)$
$M[w] \leftarrow \lfloor V_r \rfloor_l \oplus C[w-1]$
$V \leftarrow V \oplus M[w]10^* \parallel 0^c$
for $i = w-1$ **to** 1 **do**
  $V \leftarrow p_1^{-1}(V)$
  $M[i] \leftarrow C[i-1] \oplus V_r$
  $V \leftarrow C[i-1] \parallel V_c$
end
$M \leftarrow M[1]M[2]\cdots M[w]$
if $IV_c = V_c \oplus (0^{c-1} \parallel 1)$ **then**
  return M
else
  return $\perp$
end

Figure 1: The APE encryption $\mathcal{E}_K(A, M)$ and decryption $\mathcal{D}_K(A, C, T)$ algorithms for fractional messages with $w \geq 2$.

- $|M[w]| \leq r-1$ and $w = 1$ (Figs. 6 and 7). Note that the corresponding ciphertext will be of r bits. This is required for decryption to be possible;
- $|M[w]| \leq r-1$ and $w \geq 2$ (Figs. 8 and 9). Note that the ciphertext $C[w-1]$ is of size equal to $M[w]$. The reason we opt for this design property is the following: despite $M[w]$ being smaller than r bits, we require its corresponding ciphertext to be r bits for decryption to be possible. As a consequence ciphertext $C[w-1]$ is of size equal to $M[w]$;
- $|M[w]| = r$ (Figs. 10 and 11). In this special case where M is an integral message, we employ a form of '10*'-padding. Instead of occupying an extra message block for this, the usual '10*'-padding spills over into the capacity. This can be seen as an XOR of $10\cdots00$ into the capacity part of the state.

The adjustments have no influence on the decryption algorithm $\mathcal{D}$, except if $|M| \leq r$ for which a slightly more elaborate function is needed. Note that the spilling of the padding in case $|M[w]| = r$ causes security to degrade by half a bit: intuitively, APE

is left with a capacity of $c' = c - 1$ bits. We have opted for this degradation over an efficiency loss due to an additional round.

A similar spilling of the padding is also applied to the fractional associated data as indicated in Figs. 7, 9 and 11.

2.2 HANUMAN

The HANUMAN algorithm is described in Fig. 2. HANUMAN supports variable length associated data and plaintexts. As discussed in Sect. 3 we recommend the associated data and the plaintexts to be of size at most $2^{c/2}$ bits. For HANUMAN-80 this is approximately 2^{77} bytes and for HANUMAN-120 this is 2^{117} bytes. The algorithm uses two independent permutations, p_1 and p_4. The key is used twice for: (1) a part of the capacity of the initial state; and (2) after the tag truncation. A fractional input message (resp. associated data) is padded as usual by applying 10^* padding to the message (resp. associated data). In the case when $|M[w]| = r$ with M is an integral message, instead of occupying an extra message block for this, we employ the '10*' spill over into the capacity, which also can be seen as an XOR of $10 \cdots 00$ into the capacity part of the state. The encryption procedure of HANUMAN for all choices of A and M are illustrated in Figs. 12, 13, 14 and 15.

Algorithm 3: $\mathcal{E}_K(N, A, M)$

Input: $K \in \mathbf{C}^{\frac{1}{2}}$, $N \in \mathbf{C}^{\frac{1}{2}}$, $A \in \{0,1\}^*$, $M \in \{0,1\}^*$
Output: $C \in \{0,1\}^*$, $T \in \mathbf{C}^{\frac{1}{2}}$

$V \leftarrow p_1(0^r \parallel K \parallel N)$
if $A \neq \varnothing$ **then**
  $A[1]A[2] \cdots A[u] \leftarrow A$
  $A[u] \leftarrow A[u] \parallel 10^*$
  for $i = 1$ **to** $u - 1$ **do**
    $V \leftarrow p_4(A[i] \oplus V_r \parallel V_c)$
  end
  $V \leftarrow p_1(A[u] \oplus V_r \parallel V_c)$
end
$M[1]M[2] \cdots M[w] \leftarrow M$
$\ell \leftarrow |M[w]|$
$M[w] \leftarrow M[w] \parallel 10^*$
for $i = 1$ **to** w **do**
  $C[i] \leftarrow M[i] \oplus V_r$
  $V \leftarrow p_1(C[i] \parallel V_c)$
end
$C \leftarrow C[1]C[2] \cdots C[w-1]\lfloor C[w] \rfloor_\ell$
$T \leftarrow \lfloor V_c \rfloor_{\frac{c}{2}} \oplus K$
return (C, T)

Algorithm 4: $\mathcal{D}_K(N, A, C, T)$

Input: $K \in \mathbf{C}^{\frac{1}{2}}$, $N \in \mathbf{C}^{\frac{1}{2}}$, $A \in \{0,1\}^*$, $C \in \{0,1\}^*$, $T \in \mathbf{C}^{\frac{1}{2}}$
Output: $M \in \{0,1\}^*$ or $\perp$

$V \leftarrow p_1(0^r \parallel K \parallel N)$
if $A \neq \varnothing$ **then**
  $A[1]A[2] \cdots A[u] \leftarrow A$
  $A[u] \leftarrow A[u] \parallel 10^*$
  for $i = 1$ **to** $u - 1$ **do**
    $V \leftarrow p_4(A[i] \oplus V_r \parallel V_c)$
  end
  $V \leftarrow p_1(A[u] \oplus V_r \parallel V_c)$
end
$C[1]C[2] \cdots C[w] \leftarrow C$
$\ell \leftarrow |C[w]|$
for $i = 1$ **to** $w - 1$ **do**
  $M[i] \leftarrow C[i] \oplus V_r$
  $V \leftarrow p_1(C[i] \parallel V_c)$
end
$M[w] \leftarrow \lfloor V_r \rfloor_\ell \oplus C[w]$
$V \leftarrow p_1((M[w] \parallel 10^* \oplus V_r) \parallel V_c)$
$M \leftarrow M[1]M[2] \cdots M[w-1]M[w]$
$T' \leftarrow \lfloor V_c \rfloor_{\frac{c}{2}} \oplus K$
return $T = T'$? $M : \perp$

Figure 2: The HANUMAN encryption $\mathcal{E}_K(N, A, M)$ and decryption $\mathcal{D}_K(N, A, C, T)$ algorithms for fractional messages.

2.3 GIBBON

The GIBBON algorithm is described in Fig. 3. GIBBON supports variable length associated data and plaintexts. As discussed in Sect. 3 we recommend the associated data

and the plaintexts to be of size at most $2^{c/2}$ bits. For GIBBON-80 this is approximately 2^{77} bytes and for GIBBON-120 this is 2^{117} bytes. The algorithm uses three independent permutations, p_1, p_2 and p_3. The key K is used for: (1) a part of the capacity of the initial state; (2) after the initialization (first p_1 iteration); (3) before the finalization (last p_1 iteration); and (4) after the tag truncation. A fractional input message (resp. associated data) is padded as usual by applying 10^* padding to the message (resp. associated data). In the case when $|M[w]| = r$ with M an integral message, instead of occupying an extra message block for this, we employ the '10*' spill over into the capacity, which also can be seen as an XOR of $10\cdots00$ into the capacity part of the state. The encryption procedure of GIBBON for all choices of A and M are illustrated in Figs. 16, 17, 18 and 19.

Algorithm 5: $\mathcal{E}_K(N, A, M)$

Input: $K \in \mathbf{C}^{\frac{1}{2}}$, $N \in \mathbf{C}^{\frac{1}{2}}$, $A \in \{0,1\}^*$, $M \in \{0,1\}^*$
Output: $C \in \{0,1\}^*$, $T \in \mathbf{C}^{\frac{1}{2}}$

$V \leftarrow p_1\big(0^r \parallel K \parallel N\big)$
$V \leftarrow V_r \parallel (K \parallel 0^{\frac{c}{2}}) \oplus V_c$
if $A \neq \varnothing$ **then**
  $V \leftarrow p_2(V)$
  $A[1]A[2]\cdots A[u] \leftarrow A$
  $A[u] \leftarrow A[u] \parallel 10^*$
  for $i = 1$ **to** $u - 1$ **do**
    $V \leftarrow p_2\big(A[i] \oplus V_r \parallel V_c\big)$
  end
  $V \leftarrow A[u] \oplus V_r \parallel V_c$
end
$M[1]M[2]\cdots M[w] \leftarrow M$
$\ell \leftarrow |M[w]|$
$M[w] \leftarrow M[w] \parallel 10^*$
$V \leftarrow p_3\big(V\big)$
for $i = 1$ **to** w **do**
  $C[i] \leftarrow M[i] \oplus V_r$
  $V \leftarrow p_3\big(C[i] \parallel V_c\big)$
end
$V \leftarrow p_1\big(V_r \parallel (K \parallel 0^{\frac{c}{2}}) \oplus V_c\big)$
$C \leftarrow C[1]C[2]\cdots C[w-1]\lfloor C[w]\rfloor_\ell$
$T \leftarrow \lfloor V_c \rfloor_{\frac{c}{2}} \oplus K$
return (C, T)

Algorithm 6: $\mathcal{D}_K(N, A, C, T)$

Input: $K \in \mathbf{C}^{\frac{1}{2}}$, $N \in \mathbf{C}^{\frac{1}{2}}$, $A \in \{0,1\}^*$, $C \in \{0,1\}^*$, $T \in \mathbf{C}^{\frac{1}{2}}$
Output: $M \in \{0,1\}^*$ or $\perp$

$V \leftarrow p_1\big(0^r \parallel K \parallel N\big)$
$V \leftarrow V_r \parallel (K \parallel 0^{\frac{c}{2}}) \oplus V_c$
if $A \neq \varnothing$ **then**
  $V \leftarrow p_2(V)$
  $A[1]A[2]\cdots A[u] \leftarrow A$
  $A[u] \leftarrow A[u] \parallel 10^*$
  for $i = 1$ **to** $u - 1$ **do**
    $V \leftarrow p_2\big(A[i] \oplus V_r \parallel V_c\big)$
  end
  $V \leftarrow A[u] \oplus V_r \parallel V_c$
end
$C[1]C[2]\cdots C[w] \leftarrow C$
$\ell \leftarrow |C[w]|$
$V \leftarrow p_3\big(V\big)$
for $i = 1$ **to** $w - 1$ **do**
  $M[i] \leftarrow C[i] \oplus V_r$
  $V \leftarrow p_3\big(C[i] \parallel V_c\big)$
end
$M[w] \leftarrow \lfloor V_r \rfloor_\ell \oplus C[w]$
$V \leftarrow p_3\big((M[w] \parallel 10^* \oplus V_r) \parallel V_c\big)$
$M \leftarrow M[1]M[2]\cdots M[w-1]M[w]$
$V \leftarrow p_1\big(V_r \parallel (K \parallel 0^{\frac{c}{2}}) \oplus V_c\big)$
$T' \leftarrow \lfloor V_c \rfloor_{\frac{c}{2}} \oplus K$
return $T = T'$? $M : \perp$

Figure 3: The GIBBON encryption $\mathcal{E}_K(N, A, M)$ and decryption $\mathcal{D}_K(N, A, C, T)$ algorithms for fractional messages.

2.4 PRIMATE Permutation

The underlying permutation of PRIMATEs which is called PRIMATE is inspired by [8]. It has two different sizes (we write PRIMATE-80 for a 200-bit permutation and PRIMATE-120 for a 280-bit one) as well as 4 variants of each size (referred to as p_1, p_2, p_3 and p_4). PRIMATE is designed according to the wide trail strategy [11] and its structure resembles the data transform part of the Rijndael block cipher [12]. PRIMATE-80 and PRIMATE-120 operate on a 5×8 and a 7×8 state of 5-bit elements, respectively.

The first row of the state (5 bytes) is the rate of the state whereas the rest of the state is the capacity for both sizes. The state and each individual element possess big-endian encoding. PRIMATE update the internal state by means of the sequence of transformations

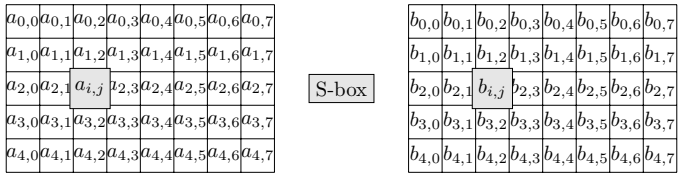
$$\mathsf{CA} \circ \mathsf{MC} \circ \mathsf{SR} \circ \mathsf{SE} \ .$$

The four permutations p_1, p_2, p_3 and p_4 of PRIMATE are defined by means of different round constants, which are generated by a 5-bit LFSR, and different number of rounds as shown in the following table.

	p_1	p_2	p_3	p_4
Number of rounds	12	6	6	12
Initial value of the LFSR	1	24	30	24

2.4.1 SubElements (SE).

The SubElements step is the only non-linear transformation of PRIMATE. It is a permutation consisting of a 5-bit S-box applied to each element of the state (shown below for PRIMATE-80).



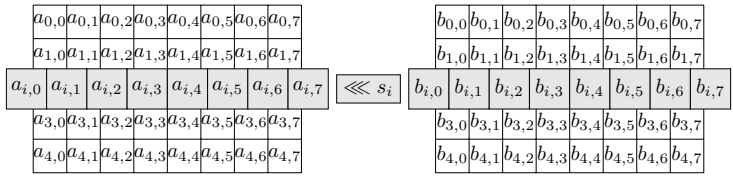
The S-box is an almost bent (AB) permutation as defined in Table 3. The maximum differential and linear probability for this S-box is 2^{-4}, which is best attainable [10] and, thus, optimal in this sense. This particular S-box has been chosen from the AB permutation set such that the area of both plain and shared implementation provide a good tradeoff, cf. [8].

Table 3: 5-bit S-box (decimal).

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S(x)	1	0	25	26	17	29	21	27	20	5	4	23	14	18	2	28
x	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
S(x)	15	8	6	3	13	7	24	16	30	9	31	10	22	12	11	19

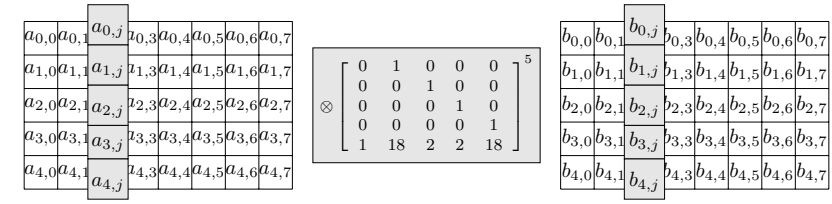
2.4.2 ShiftRows (SR).

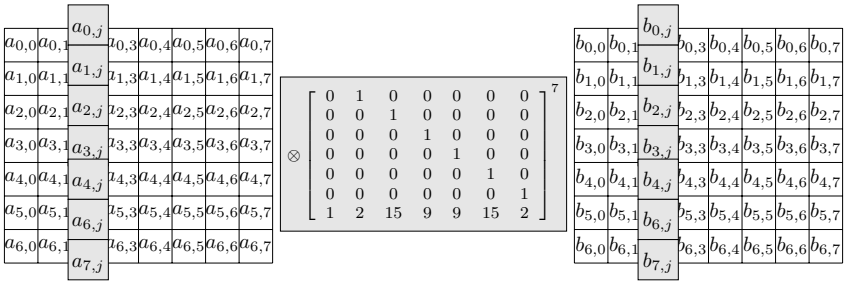
The ShiftRows step is an element transposition that cyclically shifts the rows of the state over different offsets. Row i is shifted left by $s_i = \{0, 1, 2, 4, 7\}$ positions for PRIMATE-80 (shown below) and by $s_i = \{0, 1, 2, 3, 4, 5, 7\}$ positions for PRIMATE-120. Since ShiftRows is only wiring in hardware, its overall cost is negligible.



2.4.3 MixColumns (MC).

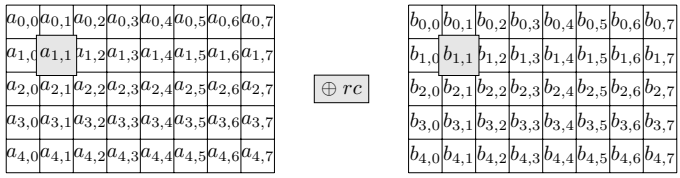
The MixColumns step is operating on the state column by column. It is a left-multiplication by a 5×5 (resp. 7×7) matrix over $\mathbb{F}_{2^5} \cong \mathbb{F}_2[x]/(x^5 + x^2 + 1)$. The main design goal of the MixColumns transformation is to follow the wide trail strategy and that it can be implemented efficiently. Therefore, we use a recursive approach [3, 14, 15, 22] to generate an MDS matrix that has a maximum (6 and 8 respectively) *branch number* (the smallest nonzero sum of active inputs and outputs of each column).





2.4.4 ConstantAddition (CA).

In this transformation the state is modified by combining the second element of the second row with a predefined constant by a bitwise XOR operation. The purpose of adding round constants is to make each round different and to break the symmetry of the other transformations. Furthermore, it provides a natural opportunity to make the parts for processing associated data and message different from each other if needed. A 5-bit Fibonacci LFSR with taps in the first (i.e. the most significant) bit and the fourth bit is used to generate the round constants rc. Therefore, the hardware implementation of ConstantAddition is in fact very cheap.



3 Security Claims

The designers claim the following levels of security, expressed in bits:

	Scheme-s	*Scheme-80*	*Scheme-120*
confidentiality of M	$c/2$	80	120
integrity of M	$c/2$	80	120
integrity of A	$c/2$	80	120
integrity of N	$c/2$	80	120

The claimed security levels correspond to the birthday bound security on the capacity of PRIMATEs-80 and PRIMATEs-120, respectively (see also Sect. 4). The security of GIBBON and HANUMAN relies on the uniqueness of the nonce, whereas APE is robust against nonce reuse. Technically, this implies that all security results of APE only hold up to common prefix: under the same associated data and nonce, two messages with the same prefix (in r-bit blocks) have the same corresponding ciphertext blocks. We refer to [1] for the technicalities.

The designers claim that APE offers certain additional security benefits. Most importantly, it is secure under the release of unverified plaintext (RUP). This means that APE is still secure if the decryption algorithm is implemented so as to output the decrypted plaintext before successful verification. This scenario arises for example when devices have insufficient memory to store the entire plaintext [13], or when the decrypted plaintext needs to be processed early due to real-time requirements [9, 21]. We refer to [2] for more information on the security under the release of unverified plaintext.

	APE-s	APE-80	APE-120
confidentiality under RUP	$c/2$	80	120
integrity under RUP	$c/2$	80	120

4 Security Analysis

PRIMATEs are indistinguishable from an ideal authenticated encryption scheme up to about $2^{c/2}$ primitive calls; implying that PRIMATEs achieve a security level of $c/2$ bits. This result is proven in the ideal model, where the underlying primitive permutations PRIMATE are assumed to be perfectly random permutations.

4.1 APE

The security results for APE can be found in [1]. APE is the first and currently the only misuse resistant permutation based authenticated encryption. The security results for APE apply *both* in the cases when nonces are unique values (full security) and also when nonces are reused (full security up to common prefix, the maximum attainable for single pass schemes). As a mode of operation for a permutation, APE is secure in the ideal model. Considering a distinguisher whose queries are of total length at most m blocks, APE is proven secure in the ideal model up to a bound of $\frac{m^2}{2^{r+c}} + \frac{2m(m+1)}{2^c}$ (integral messages) and $\frac{m^2}{2^{r+c}} + \frac{2m(m+1)}{2^{c-1}}$ (for fractional messages) [1].

We can also look at APE as a mode of operation for a block cipher where we replace the operation $(0^r \parallel K) \oplus p_1 \oplus (0^r \parallel K)$ with that of a block cipher (see [1] for a more detailed explanation). This version of APE is secure in the standard model, meaning if the underlying block cipher is a secure strong pseudorandom permutation (SPRP), then APE with a block cipher is secure as well. The bounds from the ideal model also hold in the standard model, up to twice the SPRP security of E_K. We interpret this result to mean that if $(0^r \parallel K) \oplus p_1 \oplus (0^r \parallel K)$ with p_1 instantiated by a PRIMATE is a secure SPRP, then APE with a PRIMATE is secure as well.

In the same vein, a formal security proof for APE in the case unverified plaintext is given in [2]. In more detail, the authors introduce a security model for the analysis of authenticated encryption schemes in the case when unverified plaintext is released upon decryption. In this model APE is proven to be secure upon release of unverified plaintext with no security loss (compared to the above-mentioned bounds).

Taking $c = 160$, $r = 40$ for APE-80 or $c = 240$, $r = 40$ for APE-120, the security levels approach the ones claimed in Sect. 3, but not exactly. For instance, for APE-80 we claim 80-bit security, while the proven security bound (fractional case) satisfies $\frac{m^2}{2^{r+c}} + \frac{2m(m+1)}{2^c} = \frac{1}{2}$ for $m \approx 2^{79.5}$. Similarly, for the fractional case the security bound equals $\frac{1}{2}$ for $m \approx 2^{79}$. The difference is due to the security model and proof techniques applied.

4.2 HANUMAN

HANUMAN follows a design similar to that of SpongeWrap [7]. The scheme constructs a keystream which depends on the nonce (public message number) and message, with which the message is then XORed to produce the ciphertext. As long as the nonce remains unique for each encryption, confidentiality will be achieved since the keystream will be close to uniformly random, assuming the PRIMATE permutations are close to ideal. Note that if the nonce is repeated, then the XOR of the first message blocks can be determined from the XOR of the ciphertexts. Associated data is processed via an independent permutation in order to prevent forgery attacks in which a message is first encrypted as associated data, and then again as plaintext. Attacks can be found if a collision occurs in the capacity, yet this is expected to happen only after roughly $2^{c/2}$ total queries to the underlying permutations. It is also assumed that if the verification step of the algorithm reveals that the ciphertext has been tampered with, then the algorithm returns no information beyond the verification failure.

4.3 GIBBON

The structure of GIBBON is similar to the MonkeyWrap [5] construction. The scheme generates a stream of a ciphertext and a tag depending on the nonce (public message number) and the message. Security is achieved as long as the nonce is used only once with the same key. It is also assumed that if the verification step of the algorithm reveals that the ciphertext has been tampered with, then the algorithm returns no information beyond the verification failure. In particular, no plaintext blocks are returned. A state recovery for GIBBON does not lead to trivial key recovery and also does not lead to trivial universal forgery attacks due to the key additions.

4.4 PRIMATE

This section shows some known properties of the non-linear permutation PRIMATE.

4.4.1 Differential and Linear Trails

PRIMATE has diffusion properties according to the wide trail design strategy and hence provides good bounds against differential an linear cryptanalysis. We use the technique in [19] to calculate the differential and linear hull probabilities of PRIMATE. Since the 5-bit S-box of PRIMATE is an almost bent (AB) permutation, the maximum differential and linear probability for this S-box is 2^{-4}, which provides optimal security against linear and differential cryptanalysis [10].

For PRIMATE-80, as the branch number of the linear diffusion is 6, the differential/linear probability over any two rounds does not exceed $16 \cdot (2^{-4})^6 = 2^{-20}$ and the differential/linear probability over any four rounds is upper-bounded by $(2^{-20})^5 = 2^{-100}$. For PRIMATE-120, the branch number of the linear diffusion is 8 and, therefore, the differential/linear probability over any four rounds of PRIMATE-120 is upper-bounded by $(16 \cdot (2^{-4})^8)^7 = 2^{-196}$.

This means that the probability of any twelve-round differential (and linear approximation) in PRIMATE-80 (respectively, PRIMATE-120), assuming independent rounds, does not exceed 2^{-100} (respectively, 2^{-196}). Thus, there is only a very small chance that the standard differential or linear approach would lead to a successful attack here.

4.4.2 Collision Producing Trails

Assume we have a certain difference for the message that may result in a zero difference in the state with a high probability after the difference has been injected. We call the trails corresponding to this behaviour collision producing trails. They can be used in a forgery attack on PRIMATE. Note that a linear trail of a similar shape might be used for a distinguishing attack on the keystream of PRIMATE.

The simple design of PRIMATE allows to prove good bounds against this kind of differential and linear attacks. To obtain such bounds, we adopt the mixed-integer linear programming (MILP) technique proposed in [18] to find the minimum number of differentially and linearly active S-boxes of the target ciphers. Using this technique and the optimizer CPLEX [17], we obtained the results provided in Table 4 for PRIMATE-80 and PRIMATE-120.

Table 4: Bounds for the minimum number of active S-boxes for collision producing trails of PRIMATE-80 and PRIMATE-120.

Rounds	1	2	3	4	5	6	7	8	9	10	11	12
PRIMATE-80	44	48	52	56	61	84	105	113	117	122	145	162
PRIMATE-120	63	64	72	78	113	128	143	152	177	193	209	224

For 6 rounds of GIBBON-80 (and GIBBON-120), we find that trails with at least 84 (respectively, 128) active S-boxes can produce a collision. This results in an upper bound on the differential and linear trail probability of 2^{-336} (respectively, 2^{-512}).

For 12 rounds of HANUMAN and APE, we find that a collison requires at least 162 active S-boxes of HANUMAN-80/APE-80 (224 of HANUMAN-120/APE-120 respectively). The upper bounds of the differential and linear trail probability are therefore 2^{-648} and 2^{-896} respectively.

4.4.3 Impossible Differential Cryptanalysis

Now we discuss the application of impossible differential cryptanalysis to PRIMATE. Since the branch number of PRIMATE-80 and PRIMATE-120 is 6 and 8, respectively, the number of nonzero element differences in each column before and after the MixColumns operation can never be smaller than these values. Based on this property, we construct impossible differentials for 6 and 5 rounds of PRIMATE-80 and PRIMATE-120, respectively, which are depicted in Figures 4 and 5.

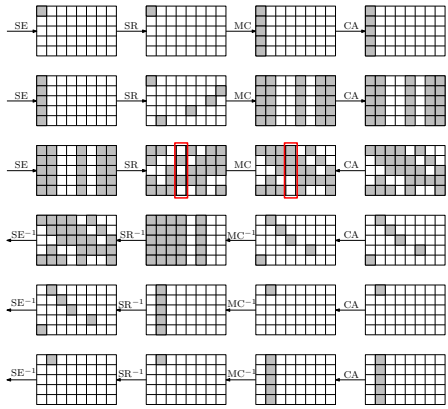


Figure 4: Impossible differential for 6 rounds of PRIMATE-80.

Taking PRIMATE-80 as an example, assume we start from the first round, if there is a nonzero difference at position (0,0) of the state, then after 2.5 rounds of PRIMATE encryption, the vector in column 3 before the MixColumns operation is $(0, *, 0, *, 0)^T$, where "*" denotes a nonzero difference. At the same time, if there is a nonzero difference in column 1 after the MixColumns of round 6 at the bottom of the distinguisher, there is a single nonzero difference at position (0,1) before MixColumns in round 5, which leads to the output vector in column 3 after MixColumns operation in round 3

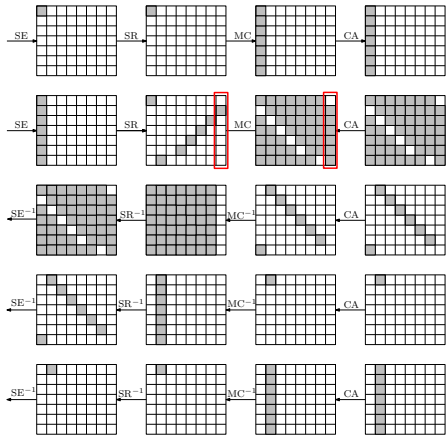


Figure 5: Impossible differential for 5 rounds of PRIMATE-120.

to be $(*, *, *, 0, 0)^T$. These columns are highlighted in red in Figure 4. This means that $M(0, *, 0, *, 0)^T = (*, *, *, 0, 0)^T$, that is, that the number of nonzero element differences before and after MixColumns is 5 which contradicts to the branch number being 6. Therefore, a 6-round impossible differential has been constructed for PRIMATE-80. Similarly, we can obtain the 5-round impossible differential of Figure 5 for PRIMATE-120.

5 Features of PRIMATEs

Permutation-based AE for lightweight applications.

The PRIMATEs authenticated encryption family is designed for *lightweight* cryptographic applications. The domain of lightweight cryptography focuses on cryptographic algorithms for extremely constrained hardware devices, where the goal is to implement an efficient cryptographic algorithm using only a very limited number of gates.

At the the core of the PRIMATEs family are the PRIMATE permutations. Since the introduction of the Sponge functions methodology [6], permutation-based cryptographic algorithms have rapidly been gaining acceptance due to their efficient implementation properties. Very recently, the sponge-based hash function Keccak [4] has been selected as the winner of the NIST SHA-3 competition.

The PRIMATE permutation is a substitution-permutation network using a 5-bit S-box with optimal linear and differential properties, and a recursive MDS matrix, which leads to a very small and efficient implementation in hardware.

Resistance against hardware side-channel attacks.

To meet the requirements of resistance against hardware side-channel attacks, the underlying permutation has been designed to offer an efficient threshold implementation

to counter first-order DPA attacks, based on glitch-free secret-sharing-based masking, cf. [8].

Online.

All PRIMATEs offer online encryption, thereby allowing the algorithm to output ciphertext blocks without the knowledge of plaintext lengths or the next plaintext blocks. PRIMATEs are inherently sequential. For lightweight applications, this is not an issue: the design goal is to use a very small number of gates, therefore parallelism would not be of any benefit.

Comparison to AES-GCM.

- GCM-AES is a block cipher based design. In comparison, PRIMATEs are smaller than similar AEAD algorithms based on a block cipher (such as AES), as our implementation does not contain a key schedule, uses smaller S-boxes (5 bits instead of 8 bits), and uses a more compact, recursive MDS matrix implementation.
- Unlike in AES-GCM, PRIMATEs handle all nonce lengths in the same way, thereby reducing the complexity of the implementation and simplifying the security analysis. We must stress, however, that when any of the parameters of the PRIMATEs (such as the nonce length) are modified, a new key must be chosen uniformly at random.
- The PRIMATEs modes avoid all the attacks that are inherent to AEAD modes based on a polynomial hash [16, 20], such as AES-GCM.

Key and Nonce Agility.

Changing the key or nonce has very little overhead for all modes in the PRIMATEs family, requiring only one permutation function call and one key XOR for GIBBON, and requires only one permutation function call for HANUMAN. In the case of APE, changing the key also requires one permutation function call. Changing the nonce requires ν/r permutation function calls.

Besides the aforementioned features that hold in general for the PRIMATEs algorithm family, several features make specific modes stand out.

Features Specific to APE, HANUMAN and GIBBON.

- APE should be used in applications where additional security is required. Like HANUMAN, APE is provably secure, based on the security of the underlying permutation. Additionally, APE provides resistance against nonce reuse [1], as well as resistance against adversaries that can observe the unverified plaintext during decryption [2]. The price to pay for this additional security is that decryption is performed backwards using the inverse of the permutation.
- HANUMAN is based on the SpongeWrap [6] design strategy. More concretely, it is the hermetic Sponge design strategy, which means that its underlying permutation should be free of any structural distinguishers.

- GIBBON is intended for lightweight applications where speed is critical and a formal security proof (based on the security of the underlying permutation) is not required. To achieve high throughput, GIBBON employs reduced-round permutations p_2 and p_3 to process the associated data and message respectively, next to the full-round permutation p_1 used for initialization and finalization.

6 Design Rationale

The PRIMATEs have been designed with lightweight hardware requirements as present in constrained devices in mind. For the mode of operation, they follow the principles of the sponge methodology, more specifically, some of the principles of SpongeWrap and MonkeyDuplex. The modes of operation are generic and free of weaknesses as justified by the formal security proofs. For the underlying permutations, the PRIMATEs follow the well-established SPN approach of Rijndael (and its wide-trail design strategy), based on almost bent S-boxes (attaining best possible differential and linear properties) as well as MDS diffusion matrices (achieving best possible differential and linear local diffusion). To favor lightweight implementations of the PRIMATEs, the MDS diffusion matrices are chosen to be recursive and the S-boxes to be 5-bit.

The PRIMATEs family includes three modes: GIBBON and HANUMAN are nonce-based, while APE has been designed to maintain security under both nonce reuse and release of unverified plaintext – scenarios that are likely to persist in highly constrained embedded systems. GIBBON does not follow the hermetic sponge-based design approach, while both HANUMAN and APE do. This allows GIBBON to be considerably faster and more energy-efficient. A state recovery for GIBBON does not lead to trivial key recovery and also does not lead to trivial universal forgery attacks due to the key additions.

The designers have not hidden any weaknesses in these ciphers.

7 Intellectual Property

The submitters are not aware of any patent involved in PRIMATEs family. Furthermore, PRIMATEs will not be patented. If any of this information changes, the submitters will promptly (and within at most one month) announce these changes on the `crypto-competitions` mailing list.

8 Consent

The submitters hereby consent to all decisions of the CAESAR selection committee regarding the selection or non-selection of this submission as a second-round candidate, a third-round candidate, a finalist, a member of the final portfolio, or any other designation provided by the committee. The submitters understand that the committee will not comment on the algorithms, except that for each selected algorithm the committee will simply cite the previously published analysis that led to the selection of the algorithm. The submitters understand that the selection of some algorithms is not a negative comment regarding other algorithms, and that an excellent algorithm

might fail to be selected simply because not enough analysis was available at the time of the committee decision. The submitters acknowledge that the committee decisions reflect the collective expert judgments of the committee members and are not subject to appeal. The submitters understand that if they disagree with published analysis then they are expected to promptly and publicly respond to those analysis, not to wait for subsequent committee decisions. The submitters understand that this statement is required as a condition of consideration of this submission by the CAESAR selection committee.

Acknowledgments. We would like to thank Miroslav Knežević for his various implementations and comments. This work was supported in part by the Research Council KU Leuven: GOA TENSE (GOA/11/007) and by the Research Fund KU Leuven, OT/13/071. In addtion, the work was supported in part by the Austrian Government through the research program COMET (Project SeCoS, Project Number 836628). Elena Andreeva and Nicky Mouha are supported by Postdoctoral Fellowships from the Flemish Research Foundation (FWO-Vlaanderen). Atul Luykx and Bart Mennink are supported by Ph.D. Fellowships from the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen). Begül Bilgin was partially supported by the FWO project G0B4213N. Qingju Wang is also funded by the Major State Basic Research Development Program of China (973 Plan) (No.2013CB338004), National Natural Science Foundation of China (No. 61073150), and Chinese Major Program of National Cryptography Development Foundation.

References

[1] Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: APE: Authenticated Permutation-Based Encryption for Lightweight Cryptography. In: FSE 2014. Lecture Notes in Computer Science, Springer (2014), to appear

[2] Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: How to Securely Release Unverified Plaintext in Authenticated Encryption. Cryptology ePrint Archive, Report 2014/144 (2014)

[3] Augot, D., Finiasz, M.: Direct Construction of Recursive MDS Diffusion Layers using Shortened BCH Codes. In: FSE 2014. Lecture Notes in Computer Science, Springer (2014), to appear

[4] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The Keccak SHA-3 submission. Submission to the NIST SHA-3 Competition (Round 3) (2011)

[5] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Permutation-Based Encryption, Authentication and Authenticated Encryption. Directions in Authenticated Ciphers (July 2012)

[6] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Cryptographic Sponge Functions, available at `http://sponge.noekeon.org/CSF-0.1.pdf`

[7] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In: Miri, A., Vaudenay, S. (eds.) Selected Areas in Cryptography 2011. Lecture Notes in Computer Science, vol. 7118, pp. 320–337. Springer (2012)

[8] Bilgin, B., Bogdanov, A., Knezevic, M., Mendel, F., Wang, Q.: Fides: Lightweight Authenticated Cipher with Side-Channel Resistance for Constrained Hardware. In: Bertoni, G., Coron, J.S. (eds.) CHES. Lecture Notes in Computer Science, vol. 8086, pp. 142–158. Springer (2013)

[9] Bogdanov, A., Mendel, F., Regazzoni, F., Rijmen, V., Tischhauser, E.: ALE: AES-based lightweight authenticated encryption. In: Moriai, S. (ed.) FSE. Lecture Notes in Computer Science, Springer (2013)

[10] Carlet, C., Charpin, P., Zinoviev, V.: Codes, Bent Functions and Permutations Suitable For DES-like Cryptosystems. Des. Codes Cryptography 15(2), 125–156 (1998)

[11] Daemen, J., Rijmen, V.: The Wide Trail Design Strategy. In: Honary, B. (ed.) IMA Int. Conf. Lecture Notes in Computer Science, vol. 2260, pp. 222–238. Springer (2001)

[12] Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer (2002)

[13] Fouque, P.A., Joux, A., Martinet, G., Valette, F.: Authenticated On-Line Encryption. In: Matsui, M., Zuccherato, R.J. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 3006, pp. 145–159. Springer (2003)

[14] Guo, J., Peyrin, T., Poschmann, A.: The PHOTON Family of Lightweight Hash Functions. In: Rogaway, P. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 6841, pp. 222–239. Springer (2011)

[15] Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.J.B.: The LED Block Cipher. In: Preneel, B., Takagi, T. (eds.) CHES. Lecture Notes in Computer Science, vol. 6917, pp. 326–341. Springer (2011)

[16] Handschuh, H., Preneel, B.: Key-Recovery Attacks on Universal Hash Function Based MAC Algorithms. In: Wagner, D. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 5157, pp. 144–161. Springer (2008)

[17] IBM: IBM ILOG CPLEX Optimizer. `http://www.ibm.com/software/integration/optimization/cplex-optimizer/`

[18] Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In: Wu, C., Yung, M., Lin, D. (eds.) Inscrypt. Lecture Notes in Computer Science, vol. 7537, pp. 57–76. Springer (2011)

[19] Park, S., Sung, S.H., Lee, S., Lim, J.: Improving the Upper Bound on the Maximum Differential and the Maximum Linear Hull Probability for SPN Structures and AES. In: Johansson, T. (ed.) FSE. Lecture Notes in Computer Science, vol. 2887, pp. 247–260. Springer (2003)

[20] Saarinen, M.J.O.: Cycling Attacks on GCM, GHASH and Other Polynomial MACs and Hashes. In: Canteaut, A. (ed.) FSE. Lecture Notes in Computer Science, vol. 7549, pp. 216–225. Springer (2012)

[21] Tsang, P.P., Solomakhin, R.V., Smith, S.W.: Authenticated streamwise on-line encryption. Dartmouth Computer Science Technical Report TR2009-640 (2009)

[22] Wu, S., Wang, M., Wu, W.: Recursive Diffusion Layers for (Lightweight) Block Ciphers and Hash Functions. In: Knudsen, L.R., Wu, H. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 7707, pp. 355–371. Springer (2012)

Diagrams

In all figures below the gray dotted box denotes the processing boundaries whenever no associated data is present.

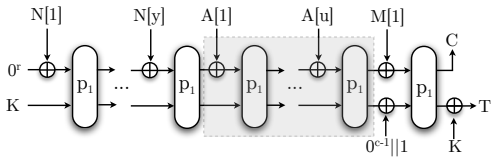


Figure 6: APE AE where $|M[w]| \leq r - 1$, $w = 1$, fractional A and padded A and M.

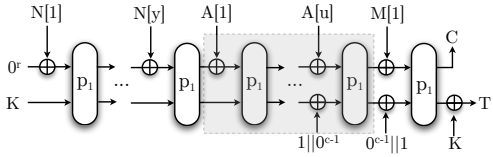


Figure 7: APE AE where $|M[w]| \leq r - 1$, $w = 1$, integral A and padded M.

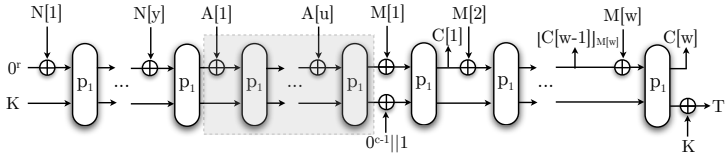


Figure 8: APE AE where $|M[w]| \leq r - 1$, $w \geq 2$, fractional A and padded A and M.

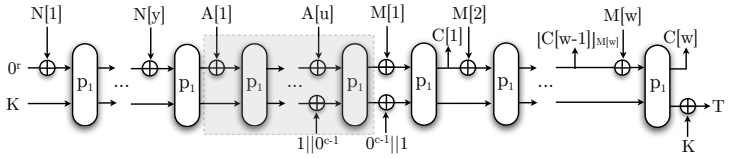


Figure 9: APE AE where $|M[w]| \leq r - 1$, $w \geq 2$, integral A and padded M.

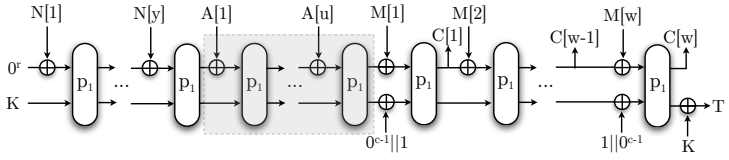


Figure 10: APE AE where $|M[w]| = r$, fractional and padded A.

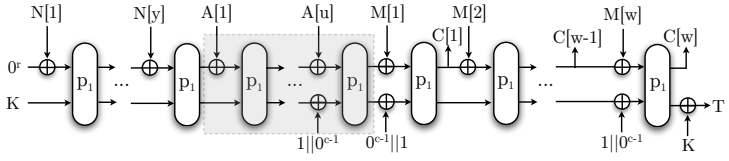


Figure 11: APE AE where $|M[w]| = r$ and integral A.

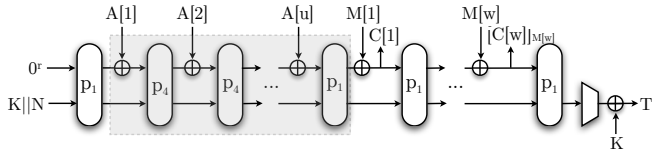


Figure 12: HANUMAN AE where $|M[w]| \leq r - 1$, fractional A and padded A and M.

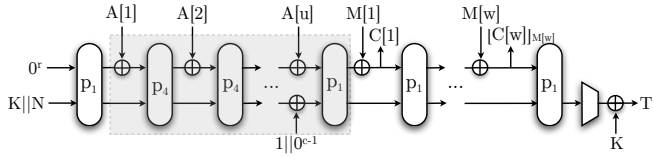


Figure 13: HANUMAN AE where $|M[w]| \leq r - 1$, integral A and padded M.

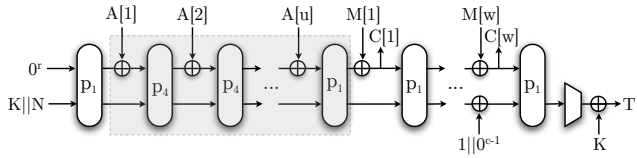


Figure 14: HANUMAN AE where $|M[w]| = r$ and fractional and padded A.

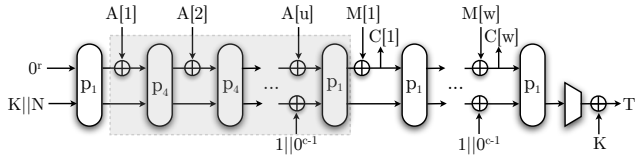


Figure 15: HANUMAN AE where $|M[w]| = r$ and integral A.

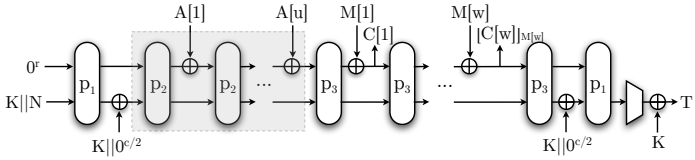


Figure 16: GIBBON AE where $|M[w]| \leq r - 1$, fractional A and padded A and M.

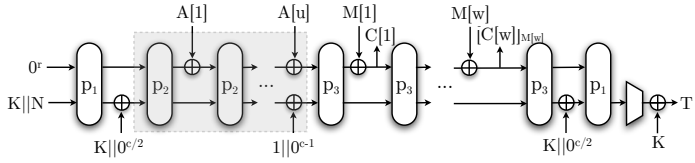


Figure 17: GIBBON AE where $|M[w]| \leq r - 1$, integral A and padded M.

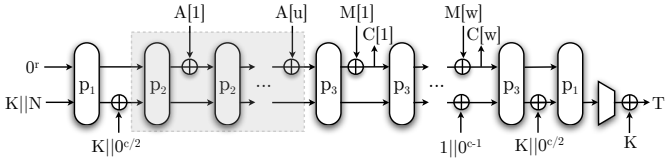


Figure 18: GIBBON AE where $|M[w]| = r$ and fractional and padded A.

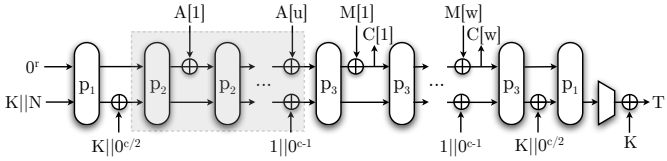


Figure 19: GIBBON AE where $|M[w]| = r$ and integral A.

Chapter 10

Improved Impossible Differential Attacks on Large-Block Rijndael

Publication Data

Q. Wang, D. Gu, V. Rijmen, Y. Liu, J. Chen, A. Bogdanov: Improved Impossible Differential Attacks on Large-Block Rijndael. In T. Kwon, M.-K. Lee, and D. Kwon (Eds.): *ICISC* 2012, volume 7839 of *Lecture Notes in Computer Science*, pages 126–140, 2013.

Contributions

Major author of the idea, the attacks and the text.

Improved Impossible Differential Attacks on Large-Block Rijndael*

Qingju Wang[1,2], Dawu Gu[1], Vincent Rijmen[2], Ya Liu[1], Jiazhe Chen[3], and Andrey Bogdanov[4]

[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China.
[2] KU Leuven, ESAT/COSIC and iMinds, Belgium.
{qingju.wang,vincent.rijmen}@esat.kuleuven.be, dwgu@sjtu.edu.cn
[3] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, School of Mathematics, Shandong University, Jinan, 250100, China.
[4] Technical University of Denmark, Department of Mathematics, Denmark.
a.bogdanov@mat.dtu.dk

Abstract. In this paper, we present more powerful 6-round impossible differentials for large-block Rijndael-224 and Rijndael-256 than the ones used by Zhang et al. in ISC 2008. Using those, we can improve the previous impossible differential cryptanalysis of both 9-round Rijndael-224 and Rijndael-256. The improvement can lead to 10-round attack on Rijndael-256 as well. With $2^{198.1}$ chosen plaintexts, an attack is demonstrated on 9-round Rijndael-224 with $2^{195.2}$ encryptions and $2^{140.4}$ bytes memory. Increasing the data complexity to 2^{216} plaintexts, the time complexity can be reduced to 2^{130} encryptions and the memory requirements to $2^{93.6}$ bytes. For 9-round Rijndael-256, we provide an attack requiring $2^{229.3}$ chosen plaintexts, 2^{194} encryptions, and $2^{139.6}$ bytes memory. Alternatively, with $2^{245.3}$ plaintexts, an attack with a reduced time of $2^{127.1}$ encryptions and a memory complexity of $2^{90.9}$ bytes can be mounted. With $2^{244.2}$ chosen plaintexts, we can attack 10-round Rijndael-256 with $2^{253.9}$ encryptions and $2^{186.8}$ bytes of memory.

Keywords: block cipher, impossible differential attack, Rijndael, large block

1 Introduction

Rijndael [11] is a block cipher designed by Joan Daemen and Vincent Rijmen built upon a Substitution Permutation Network (SPN). A subset of Rijndael variants has been standardized as Advanced Encryption Standard (AES) by the U.S. National Institute of Standards and Technology (NIST) [14] in 2002.

* This work was supported by the National Natural Science Foundation of China (No. 61073150), and in part by the Research Council K.U.Leuven: GOA TENSE, the IAP Program P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and in part by the European Commission through the ICT program under contract ICT-2007-216676 ECRYPT II.

Rijndael follows the design principles of Square [9]. In its full version, both the block and the key sizes can range from 128 to 256 bits in steps of 32 bits. For AES, the block size of Rijndael is restricted to 128 bits. This paper deals with non-AES Rijndael variants, that is, *large-block Rijndael-b*, with $b > 128$ indicating the block size and key size in bits.

AES is probably the most well-studied block cipher, having received about 15 years of extensive public scrutiny by now. Square attacks, impossible differential attacks, boomerang attacks, rectangle attacks and meet-in-the-middle attacks in both the single-key and related-key settings are just several prominent examples of cryptanalysis techniques applied to AES [1, 4–7, 12, 17, 20, 22–24, 26–28].

The large-block Rijndael is arguably less analyzed, being a highly relevant cipher though. Among others, an important motivation for the study of large-block Rijndael is the deployment of Rijndael-like permutations in the design of hash functions, Whirlwind [2] and SHA-3 finalist Grøstl [16] constituting some especially interesting instances. We mention here several multiset and integral cryptanalytic results [13, 15, 18, 21], as well as impossible differential cryptanalysis [19, 25]. In terms of the impossible differential cryptanalysis – the major object of our study in this paper – the best attack has been proposed by Zhang et al. [25] which cryptanalyzes 9-round Rijndael-224 and Rijndael-256 with 2^{209} and $2^{208.8}$ encryptions, respectively.

Impossible differential cryptanalysis, which was proposed by [3,8], is a widely used cryptanalytic technique. The attack starts with finding a certain input difference that can never result in a certain output difference, which makes up an *impossible differential*. Usually, impossible differentials have truncated input and output differences. By adding rounds before and/or after the impossible differential, one can collect pairs with certain plaintext and ciphertext differences. If there exists a pair that meets the input and output values of the impossible differential under some subkey bits, these bits must be wrong. In this way, we discard as many wrong keys as possible and exhaustively search the rest of the keys. The early abort technique is usually used during the key recovery phase, that is, one does not guess all the subkey bits at once, but guesses some subkey bits instead to discard some pairs that do not satisfy certain conditions step by step. In this case, we can discard the unwished pairs as soon as possible to reduce the time complexity.

Our Contributions. In this paper, we present more powerful 6-round impossible differentials for Rijndael-224 and Rijndael-256. Using these impossible differentials, we can improve the existing impossible differential cryptanalyses of both Rijndael-224 and Rijndael-256. In addition, the improvement can result in a 10-round attack on Rijndael-256.

Our impossible differentials for both Rijndael-224 and Rijndael-256 have more active bytes in the output difference and, therefore, the number of subkey bytes needed to be guessed during the key recovery phase can range with more options, while the probability for a pair of plaintexts to pass the test of sieving wrong pairs is higher compared to [25].

For 9-round Rijndael-256, utilizing the new impossible differential and depending on the number of subkey bytes needed to be guessed in key recovery

Table 1. Summary of Attacks on Rijndael-224 and Rijndael-256

Cipher	Number of Round	Complexity Time (EN)	Complexity Data(CP)	Attack type	Source
Rijndael-224	7	2^{141}	$2^{130.5}$	Multiset	[18]
	7	2^{167}	2^{138}	Imp. Diff.	[19]
	7	$2^{113.4}$	$2^{93.2}$	Imp. Diff.	[25]
	9	$2^{196.5}$	$2^{196.5}$	Integral	[21]
	9	2^{209}	$2^{212.3}$	Imp. Diff.	[25]
	9	$2^{195.2}$	$2^{198.1}$	Imp. Diff.	sect. 4
	9	2^{162}	2^{208}	Imp. Diff.	sect. 4
	9	2^{130}	2^{216}	Imp. Diff.	sect. 4
Rijndael-256	7	$2^{128}-2^{119}$	$2^{128}-2^{119}$	Part. Sum	[13]
	7	2^{141}	$2^{130.5}$	Multiset	[18]
	7	2^{44}	6×2^{32}	Integral	[15]
	7	2^{182}	2^{153}	Imp. Diff.	[19]
	7	$2^{113.2}$	2^{93}	Imp. Diff.	[25]
	8	$2^{128}-2^{119}$	$2^{128}-2^{119}$	Integral	[15]
	9	2^{204}	$2^{128}-2^{119}$	Integral	[15]
	9	$2^{174.5}$	$2^{132.5}$	Integral	[21]
	9	$2^{208.8}$	$2^{244.3}$	Imp. Diff.	[25]
	9	2^{194}	$2^{229.3}$	Imp. Diff.	subsect. 3.2
	9	$2^{159.1}$	$2^{237.3}$	Imp. Diff.	subsect. 3.3
	9	$2^{127.1}$	$2^{245.3}$	Imp. Diff.	subsect. 3.3
	10	$2^{253.9}$	$2^{244.2}$	Imp. Diff.	subsect. 3.4

CP: Chosen Plaintext; EN: Number of round encryptions

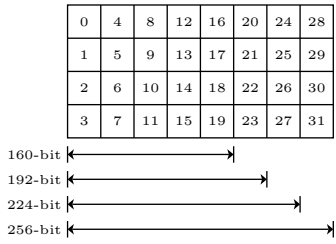
phase, three improved attacks can be obtained. If we guess the same number of subkey bytes as [25], an attack can be mounted with reduced data complexity of $2^{229.3}$ Chosen Ciphertexts (CP), time complexity 2^{194} encryptions and memory complexity $2^{139.6}$ bytes respectively. In addition, if the number of subkey bytes need to guess is less than [25], given $2^{237.3}$ CP, we can attack 9-round Rijndael-256 with $2^{159.1}$ encryptions and $2^{115.3}$ bytes of memory. If the data complexity are increased to $2^{245.3}$ CP, the time and memory complexity can be significantly reduced to $2^{127.1}$ encryptions and $2^{90.9}$ bytes. Moreover, based on the same impossible differential, considering $2^{244.2}$ CP, we can even attack 10-round Rijndael-256 with $2^{253.9}$ encryptions and $2^{186.8}$ bytes of memory accesses. As for Rijndael-224, similarly three attacks can also be mounted on 9-round with lower complexity. With $2^{198.1}$ CP, an attack is demonstrated on 9-round Rijndael-224 with $2^{195.2}$ encryptions and $2^{140.4}$ bytes memory. Take 2^{208} CP, we can attack 9-round Rijndael-224 with 2^{162} encryptions and 2^{117} bytes memory. Increasing the data complexity to 2^{216} chosen plaintexts, the time complexity can be greatly reduced to 2^{130} encryptions and the memory requirements to $2^{93.6}$ bytes.

To the best of our knowledge, these results are the best impossible differential attacks on Rijndael-224 and Rijndael-256. We summarize our results for Rijndael-224 and Rijndael-256, as well as the major previous results in Table 1.

The remainder of this paper is organized as follows. Section 2 gives a brief overview of Rijndael and introduces the notations used in this paper. In Section 3 we first derive a new 6-round impossible differential, and then present three improved impossible differential attacks on 9-round Rijndael-256. The attack can also be extended to 10-round Rijndael-256. Then in Section 4, after a new 6-round impossible differential distinguisher is presented, we mount three improved attacks on 9-round Rijndael-224. Finally, we conclude this paper in Section 5.

2 Description of Rijndael and Notations

Rijndael has N_r rounds, which can be 10, 12, or 14 depending on the key size. In Rijndael, both the text block and the key sizes can range for 128 up to 256 bits in steps of 32 bits. The 128-bit block version of Rijndael, with the key size 128, 192 or 256, is officially known as AES [14]. The plaintext, ciphertext, subkey, and all the intermediate data are represented by a $4 \times N_b$ `state matrix` of bytes, where N_b is the number of 32-bit words in the block. The byte indexing for the `state matrix` is shown in the left part of Figure 1. The key schedule



N_b	C_0	C_1	C_2	C_3
5	0	1	2	3
6	0	1	2	3
7	0	1	2	4
8	0	1	3	4

Fig. 1. Byte Index of the State Matrix and the Shift Offsets for Each Block Length N_b

derives $(N_r + 1)$ b-bit `RoundKey(RK)` from the master key, denoted from RK_0 to RK_{N_r}. The Expanded Key is a linear array of 4-byte words and is denoted by $W[N_b * (N_r + 1)]$. The first N_k words $W[0]\|W[1]\| \cdots \|W[N_k - 1]$ are directly initialised by the N_k words of the master key, while the remaining key words, $W[i]$ for $i \in \{N_k, \cdots, N_k*(N_r+1)-1\}$ are generated by the following algorithm:

`if` $(i \bmod N) = 0$ `then` $W[i] = W[i - N_k] \oplus f(W[i-1]) \oplus Rcon[i/N_k]$
`else if` $((N_k > 6)$ and $(i \bmod N = 4))$ `then` $W[i] = W[i-N_k] \oplus g(W[i-1])$
`else` $W[i] = W[i - N_k] \oplus W[i-1]$

where $f, g : \{0,1\}^{32} \to \{0,1\}^{32}$ are nonlinear permutations, *Rcon* denotes fixed constants depending on its input. `Roundkey` RK_i is given by the Round Key buffer words $W[N_b * i]$ to $W[N_b * (i+1)]$.

The round function, which is repeated $(N_r - 1)$ times, involves four operations: `SubBytes(SB)`, `ShiftRows(SR)`, `MixColumns(MC)` and `AddRoundKey(ARK)`.

The `SubBytes` operation consists of the parallel application of a fixed 8-bit to 8-bit Sbox to each byte of the state. `ShiftRows` is a byte transposition that left shifts the rows of the state over different offsets. The shift offsets C_i of row i which depend on the block length N_b, are specified in the right part of Figure 1 for each block length of Rijndael. `MixColumns` is an (4×4) Maximum Distance Separable (MDS) matrix multiplication over $GF(2^8)$ for each column of the state. Obviously the branch number of this MDS matrix is five. `AddRoundKey` consists of the exclusive-or combination of the `RoundKey` with the intermediate state.

These $(N_r - 1)$ rounds are surrounded by an whitening layer consisting of `AddRoundKey` only, and the last round with `MixColumns` operation omitted. We also assume that this is the same case for the reduced Rijndael we are focusing on throughout this paper. Here we only give a brief description of Rijndael, for more detailed specification of the cipher, we refer to [10, 11].

We will also use the technique that the operations of `MixColumns` and `AddRoundKey` can be interchanged under some conditions [11]. Here we introduce some notations as well for later use in the following.

X_i : the state of the i-th round;
ΔX_i : the difference for state of the i-th round;
X_i^I : the input state of the i-th round;
RK_i : the subkey of the i-th round;
RK_i^* : the value of the subkey of the i-th round after the inverse of the `MixColumns` operation;
X_i^{SB} : the intermediate state after the `SubBytes` operation in the i-th round;
X_i^{SR} : the intermediate state after the `ShiftRows` operation in the i-th round;
X_i^{MC} : the intermediate state after the `MixColumns` operation in the i-th round;
X_i^W : the intermediate state after the `AddRoundKey` operation with RK_i^* in the i-th round;
X_i^O : the intermediate state after the `AddRoundKey` operation in the i-th round;
? : an indeterminate difference.

Obviously, $X_i^I = X_{i-1}^O$ hold. Note that the operation of `AddRoundKey` will be represented as ARK^* throughout this paper when the Roundkey RK^* is used.

3 Improved Impossible Differential Attacks on Rijndael-256

In this section, we first give a new 6-round impossible differential for Rijndael-256 in Section 3.1. Based on this impossible differential and depending on the number of the subkey bytes need to guess during the key recovery phase, three improved 9-round impossible differential attacks compared to [25] will be presented in Subsection 3.2 and 3.3 respectively. Using the same impossible differential, we can extend it to an attack of 10-round Rijndael-256 in Subsection 3.4.

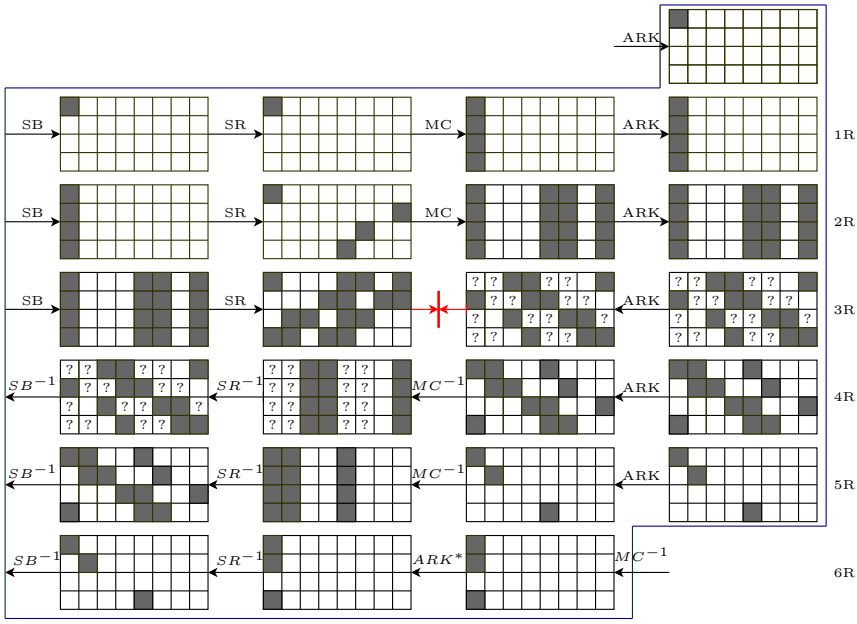


Fig. 2. The New 6-Round Impossible Differential of Rijndael-256

3.1 New 6-Round Impossible Differential on Rijndael-256

Assume we start with round 1 (denoted as 1R in Figure 2) and the input difference ΔX_1 has one active byte whereas the other bytes are zero, one illustration with the first byte active is depicted in Figure 2. Then request 2.5 rounds encryption from the `SB` operation in round 1 to the `SR` operation in round 3 to get the difference ΔX_3^{SR}. Consider the output difference with three nonzero bytes in the first column of the state, one option with the active bytes at (0,1,3) is shown in Figure 2, the other option has the active bytes at (0,2,3). Decrypt 3.5 rounds (as depicted from the operation `ARK`* in round 6 to the operation `ARK` in round 3) in order to get the difference ΔX_3^{MC}. Note there is no `AddRoundKey` operation in round 6 because the order of `MixColumns` and `AddRoundKey` operations can be interchanged as mentioned before in Section 2. For the third column of the state ΔX_3^{SR}, the number of nonzero bytes is one, while it is at most three for the nonzero bytes of ΔX_3^{MC} (it is indeterminate at byte 9). Since the branch number of the MDS matrix is five, there is a contradiction before and after the `MixColumns` operation. By similar reasoning, there is also a contradiction in the seventh column in the state before and after the `MixColumns` operation in round 3. Therefore, we make up a 6-round impossible differential for Rijndael-256.

There exist more active bytes in the output of the impossible differential compared to [25] (they has one byte), therefore we have more options in guessing the subkey bytes to meet the output of the impossible differential while adding extra rounds after the impossible differential distinguisher. There are three active

bytes in one column after MC^{-1} at the bottom of the impossible differential, thus the number of the subkey bytes we need to guess in order to calculate the output after MC^{-1} can range from two to four, therefore three attacks can be mounted using this impossible differential.

3.2 9-Round Attack on Rijndael-256 with Lower Data Complexity

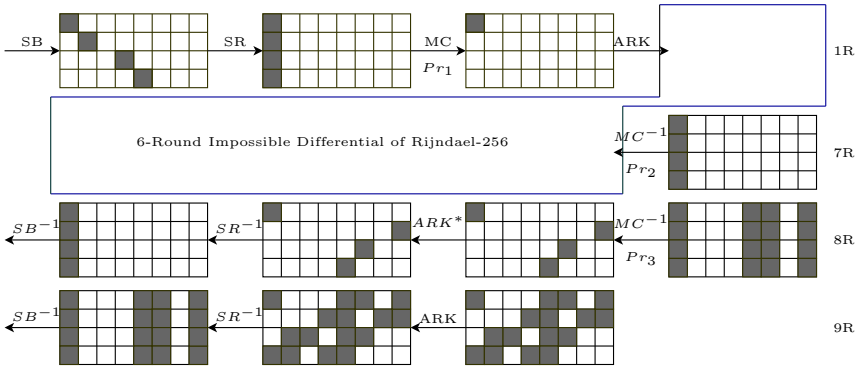


Fig. 3. Improved 9-Round Attack on Rijndael-256 with Lower Data Complexity

In this subsection we present the attacks on 9-round Rijndael-256 utilizing the 6-round impossible differential in Subsection 3.1. In our attack, we guess the same number (i.e. four) of subkey bytes of RK_8^* as [25] in the *key recovery phase*. As a result, 16 bytes of subkey RK_9 will have to be guessed to partially decrypt round 9 in order to calculate X_8^W (as shown in Figure 3), which will also provide a 128-bit condition for ciphertexts in the *data collection phase*. The number of active bytes at the end of the new impossible differential distinguisher will filter out more wrong pairs during the *key recovery phase*. Therefore, an improved attack with significantly reduced data complexity compared to [25] will be result in. The detailed procedures of the attack will be described as follows.

Data Collection. We first choose 2^n structures of plaintexts. In each structure the plaintexts range over all 32-bit values at bytes (0,5,14,19), while the other bytes can take certain fixed values. Each structure includes about $(2^{32})^2/2 = 2^{63}$ pairs of plaintexts, therefore $2^n \cdot 2^{63} = 2^{n+63}$ pairs of plaintexts will be prepared. Encrypt these pairs and keep the one whose ciphertext difference are zero at bytes (1,2,4,5,8,9,11,12,14,21,23,24,26,27,30,31). The probability of such ciphertexts is about $2^{-8 \cdot 16} = 2^{-128}$, thus the expected number of the remaining pairs after this phase is about $2^{n+63-128} = 2^{n-65}$.

The sieving of the ciphertexts can be done by birthday attack. As a result, the time complexity of this phase is about 2^{n+32}. In addition, we need $2^{n-65} \cdot 5 \cdot 32 = 2^{n-57.7}$ bytes memory to store these pairs.

Key Recovery. In order to check if the pairs generated in *data collection phase* satisfy the impossible differential in Figure 3, we need to guess certain bytes of subkey $(RK_9, RK_0, , RK_8^*)$ during the *key recovery phase*. The details are described in the following:

Step 1 For all the pairs of plaintext obtained in the data collection phase, we guess the 32-bit subkey $(RK_{9,0}, RK_{9,29}, RK_{9,22}, RK_{10,19})$ and partially decrypt round 9 to compute the first column of ΔX_8^W. Check if the differences at byte (1,2,3) are zero. If it is not the case, discard the pair. The probability of this event is 2^{-24}. After this step the expected number of remaining pairs is about $2^{n-65-24} = 2^{n-89}$.

Step 2 For every guess of the 32-bit subkey $(RK_{9,16}, RK_{9,13}, RK_{9,6}, RK_{9,3})$, we partially decrypt round 9 to compute the fifth column of ΔX_8^W. Check if the differences at byte (0,1,2) are zero. If it is not the case, discard the pair. The probability of this event is 2^{-24}. After this step the expected number of remaining pairs is about $2^{n-89-24} = 2^{n-113}$.

Step 3 For every guess of the 32-bit subkey $(RK_{9,20}, RK_{9,17}, RK_{9,10}, RK_{9,7})$, we partially decrypt round 9 to compute the sixth column of ΔX_8^W. Check if the differences at byte (0,1,3) are zero. If it is not the case, discard the pair. The probability of this event is 2^{-24}. After this step the expected number of remaining pairs is about $2^{n-113-24} = 2^{n-137}$.

Step 4 For every guess of the 32-bit subkey $(RK_{9,28}, RK_{9,25}, RK_{9,18}, RK_{9,15})$, we partially decrypt round 9 to compute the eight column of ΔX_8^W. Check if the differences at byte (0,2,3) are zero. If it is not the case, discard the pair. The probability of this event is 2^{-24}. After this step the expected number of remaining pairs is about $2^{n-137-24} = 2^{n-161}$.

Step 5 We need to guess the 32-bit of subkey $(RK_{0,0}, RK_{0,5}, RK_{0,14}, RK_{0,19})$ for all the remaining pairs, and partially encrypt round 1 to get the first column of ΔX_1^{MC}. Check if the difference at byte (1,2,3) are zero. If it is not the case, discard the pair. The probability of this event is about $4 \cdot (2^8 - 1)/2^{32} \approx 2^{-22}$. Thus after this step the remained pairs is about $2^{n-161-22} = 2^{n-183}$.

Step 6 For every guess of the 16-bit subkey $(RK_{8,0}^*, RK_{8,29}^*, RK_{8,22}^*, RK_{8,19}^*)$, partially decrypt round 8 to compute the first column of ΔX_7^W. Check if the differences at the third byte is zero. If it is correct, delete all the 32-bit subkey guesses of RK_8^* since such a differential is impossible, each subkey guess that proposes such a differential is a wrong key. After analyzing all the 2^{n-183} remaining pairs, if there still remains value of RK_8^*, output the 192-bit subkey guess of (RK_0, RK_8^*, RK_9) as the correct key. Our experiments provide the evidence that the probability of the pairs pass this step is about $Pr_2 = 2 \cdot 2^{-8} = 2^{-7}$.

The process steps of the *key recovery phase* above are described in Table 2, whereas the second column lists the bytes need to be guessed in the correspond-

ing round for each step. The third column stands for the number of remained pairs after sieving in each step, and the time complexity of each step will be measured in the fourth column in Table 2. Note that when evaluating the time complexity of the recovery, it is measured by one round encryption. Similar tables will be adopted to describe the steps of the *key recovery phase* throughout this paper.

Table 2. Key Recovery Processes of the Attack on Rijndael-256 with lower Data Complexity

Step	Guessed Bytes	#Pairs Kept	Time Complexity
1	$RK_9 : 0, 29, 22, 19$	$2^{n-65-24} = 2^{n-89}$	$2^{32} \cdot 2^{n-65} \cdot 2/8 = 2^{n-35}$
2	$RK_9 : 16, 13, 6, 3$	$2^{n-89-24} = 2^{n-113}$	$2^{64} \cdot 2^{n-89} \cdot 2/8 = 2^{n-27}$
3	$RK_9 : 20, 17, 10, 7$	$2^{n-113-24} = 2^{n-137}$	$2^{96} \cdot 2^{n-113} \cdot 2/8 = 2^{n-19}$
4	$RK_9 : 28, 25, 18, 15$	$2^{n-137-24} = 2^{n-161}$	$2^{128} \cdot 2^{n-137} \cdot 2/8 = 2^{n-11}$
5	$RK_0 : 0, 5, 14, 19$	$2^{n-161-22} = 2^{n-183}$	$2^{160} \cdot 2^{n-161} \cdot 2/8 = 2^{n-3}$
6	$RK_8^* : 0, 29, 22, 19$	-	$2^{192} \cdot 2 \cdot [1 + (1-2^{-7}) + (1-2^{-7})^2 + \cdots + (1-2^{-7})^{2^{n-183}}]/8$

Analysis of the Attack Take $n = 197.3$, after analyzing all the remaining pairs, there will be about $2^{192} \cdot (1-2^{-7})^{2^{n-183}} = 2^{-36.2}$ wrong subkeys of RK_0 left, we can get rid of the wrong subkeys by $2^{187.8}$ trail encryptions. Therefore the data complexity will be $2^{n+32} = 2^{229.3}$, the time complexity will be $2^{197.2}/9 \approx 2^{194}$ 9-round encryptions, the memory required is about $2^{139.6}$ bytes.

3.3 9-Round Attack on Rijndael-256 with lower Time Complexity

We will use the same new 6-round impossible differential as the previous section, which helps to get rid of more pairs. As mentioned in Subsection 3.1, the number of subkey bytes need to be guessed in round 8 can be reduced compared to [25], i.e. two or three bytes of RK_8^*. Here we take two bytes for example. As a result, it will be the same case for round 9, which means fewer columns need to be decrypted. Meanwhile, 192 bits are zero for the ciphertexts, which provides an stronger condition of ciphertexts for sieving wrong pairs compared to 128 bits in [25]. As a result an improved attack with the time complexity greatly reduced can be mounted on 9-round Rijndael-256. Because of the similarity of the attack with the one in Subsection 3.2, only a brief description of this attack will be demonstrated as follows:

In the *data collection phase*, we take the same structures as in Subsect 3.2, thus 2^{n+63} pairs of plaintexts will be generated. there exists the 192-bit condition for ciphertext to discard wrong pairs, thus the expected number of the remaining pairs is $2^{n+63-192} = 2^{n-129}$ at the end of this phase.

In the *key recovery phase*, we only guess 8 bytes of RK_9, 4 bytes of RK_0 and 2 bytes of RK_8^* to check if the impossible differential will be satisfied for the

remaining pairs. When filtering out wrong pairs, we obtain the probabilities that the pairs pass the tests in round 8, round 1 and round 7 are $Pr_3 = (2^{-24})^2 = 2^{-48}$, $Pr_1 = 4 \cdot (2^8 - 1)/2^{32} \approx 2^{-22}$ and $Pr_2 = 2 \cdot 2^{-8} \approx 2^{-7}$ respectively. The expected number of remaining pairs after this phase is about 2^{n-199}. The steps and the time complexity evaluation of this phase are given in Table 3. Take

Table 3. Key Recovery Processes of the Improved Attack on Rijndael-256 with lower Time Complexity

Step	Guessed Bytes	#Pairs Kept	Time Complexity
1	$RK_9 : 0, 29, 22, 19$	$2^{n-129-24} = 2^{n-153}$	$2^{32} \cdot 2 \cdot 2^{n-129}/8 = 2^{n-99}$
2	$RK_9 : 28, 25, 18, 15$	$2^{n-153-24} = 2^{n-177}$	$2^{64} \cdot 2 \cdot 2^{n-153}/8 = 2^{n-91}$
3	$RK_0 : 0, 5, 14, 19$	$2^{n-177-22} = 2^{n-199}$	$2^{96} \cdot 2 \cdot 2^{n-177}/8 = 2^{n-83}$
4	$RK_8^* : 0, 29$	-	$2^{112} \cdot 2 \cdot [1 + (1 - 2^{-7}) + (1 - 2^{-7})^2 + \cdots + (1 - 2^{-7})^{2^{n-199}}]/16$

$n = 213.3$, the data complexity is $2^{n+32} = 2^{245.3}$ CP, the time complexity will be $2^{130.3}/9 \approx 2^{127.1}$ 9-round encryptions, the memory required is about $2^{90.9}$ bytes.

Moreover, as mentioned at the beginning of this subsection, it is also possible to guess three bytes of the subkey RK_8^* to calculate ΔX_7^W in order to check if the impossible differential can be satisfied. As a result 12 bytes of RK_9 have to be guessed to partially decrypt round 9 in the key recovery phase. In this case, the data complexity is about $2^{237.3}$ CP, the time complexity is about $2^{159.1}$ 9-round encryption, and the memory is about $2^{115.3}$ bytes.

3.4 10-Round Impossible Differential Attack on Rijndael-256

Based on the same impossible differential as in the previous subsection, we will extend two rounds backwards and forwards respectively, an attack on 10-round Rijndael-256 will be led with complexity less than exhaustive search. We adopt the 9-round attack with lower time complexity in Subsection 3.3 to act as our internal 9-round attack, on which we make some modification. In addition, we will take the key schedule into consideration. The brief attack will be given out as follows.

In the *data collection phase*, take 2^n structures of plaintexts, in which the plaintexts range over 128-bit values at bytes (0,3,4,5,9,12,14,16~19,21,23,26,30,31), while the other bytes can take certain fixed values. Each structure includes about $(2^{128})^2/2 = 2^{255}$ pairs of plaintexts, therefore $2^n \cdot 2^{255} = 2^{n+255}$ pairs of plaintexts are obtained. Encrypt these pairs and keep the one whose ciphertext difference are zero at bytes (1~14,16,17,20,21,23,24,26,27,30,31). The probability of such ciphertexts is about $2^{-8 \cdot 24} = 2^{-192}$, thus the expected number of the remaining pairs after this phase is about $2^{n+255-192} = 2^{n+63}$.

In the *key recovery phase*, as in the 9-round attack in Subsection 3.3, 8 subkey bytes of RK_{10} should be guessed. Because of the extra round backward extension,

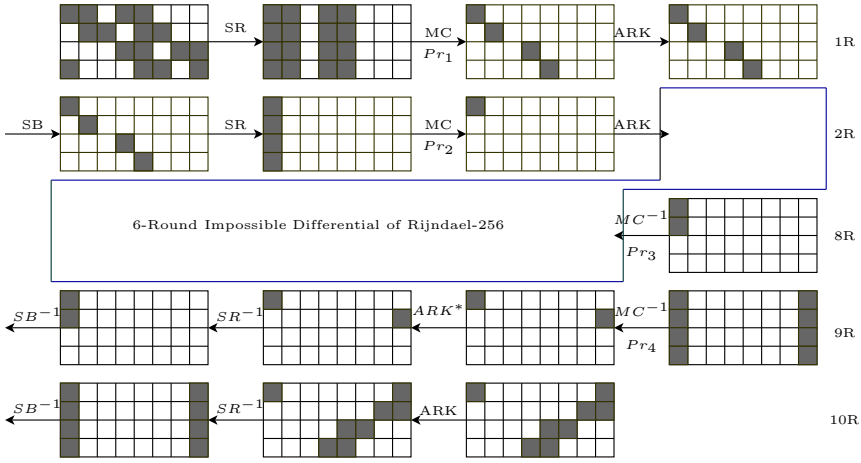


Fig. 4. 10-Round Impossible Differential Attack on Rijndael-256

16 bytes of RK_0 and 4 bytes of RK_1 will also be guessed respectively. At the end of this phase, the number of remaining pairs is 2^{n-103}. The process steps of this phase are described in Table 4. By the key schedule, we can calculate

Table 4. Key Recovery Processes of the Attack on 10-Round Rijndael-256

Step	Guessed Bytes	#Pairs Kept	Time Complexity
1	$RK_{10} : 0, 29, 22, 19$	$2^{n+63-24} = 2^{n+39}$	$2^{32} \cdot 2^{n+63} \cdot 2/8 = 2^{n+93}$
2	$RK_{10} : 28, 25, 18, 15$	$2^{n+39-24} = 2^{n+15}$	$2^{64} \cdot 2^{n+39} \cdot 2/8 = 2^{n+101}$
3	$RK_0 : 0, 5, 14, 19$	$2^{n+15-24} = 2^{n-9}$	$2^{96} \cdot 2^{n+15} \cdot 2/8 = 2^{n+109}$
4	$RK_0 : 4, 9, 18, 23$	$2^{n-9-24} = 2^{n-33}$	$2^{128} \cdot 2^{n-9} \cdot 2/8 = 2^{n+117}$
5	$RK_0 : 12, 17, 26, 31$	$2^{n-33-24} = 2^{n-57}$	$2^{160} \cdot 2^{n-33} \cdot 2/8 = 2^{n+125}$
6	$RK_0 : 16, 21, 30, 3$	$2^{n-57-24} = 2^{n-81}$	$2^{192} \cdot 2^{n-57} \cdot 2/8 = 2^{n+133}$
7	$RK_1 : 0, 5, 14, 19$	$2^{n-81-22} = 2^{n-103}$	$2^{224} \cdot 2^{n-81} \cdot 2/8 = 2^{n+141}$
8	$RK_9^* : 0, 29$	-	$2^{240} \cdot 2 \cdot [1 + (1-2^{-7}) + (1-2^{-7})^2 + \cdots + (1-2^{-7})^{2^{n-103}}]/16$

$RK_{0,29}$ from $RK_{0,0}$ and $RK_{1,0}$. $RK_{0,5}$ and $RK_{1,5}$ determine $RK_{1,1}$, then $RK_{1,1}$ together with $RK_{0,30}$ determine $RK_{0,1}$. Therefore, in order to recover the key, there are 14 bytes of RK_0 left to guess. We can take $n = 116.2$, from the *data collection phase* we know that the data complexity of the attack is $2^n \cdot 2^{128} = 2^{244.2}$ Chosen Ciphertext (CP). In the *key recovery phase*, after analyzing the remaining $2^{n-103} = 2^{13.2}$ pairs, the expected number of wrong subkeys is $2^{240} \cdot (1-2^{-7})^{2^{n-103}} \approx 2^{133.5}$. With about $2^{112} \cdot 2^{133.5} = 2^{245.5}$ trail encryptions, the correct key will be recovered. The time complexity is about $2^{257.2}/10 \approx 2^{253.9}$ 10-round encryptions. The memory required to store the pairs is about $2^{186.8}$ bytes.

4 Improved Impossible Differential Attacks on Rijndael-224

In this section, we first give a new 6-round impossible differential of Rijndeal-224 (see Figure 5 in Appendix A). Utilizing the new 6-round impossible differential, we extend one round at the top and two rounds at the bottom to mount 9-round impossible differential attacks on Rijndael-224. As we can see there exist three active bytes at the bottom of the distinguisher, as a result the number of the subkey bytes need to guess in round 8 during the key recovery stage can range from two to four. Therefore three 9-round attacks on Rijndael-224 can be obtained respectively. First assume there are four bytes of subkey RK_8^* need to guess in order to check if the impossible differential distinguisher is satisfied during the key recovery phase, as depicted in Figure 6.

In the *data collection phase*, choose structures of 2^{32} plaintexts, in which the plaintexts take all possible 32-bit values at bytes (0,5,10,19) while the others take certain fixed values. Take $2^{166.1}$ structures, about $2^{229.1}$ pairs of plaintexts will be generated. Filter out the pairs whose ciphertext difference are not zero at byte (1∼5,8,10,13,16,19,23,26). Because of this 96-bit condition for ciphertexts, the expected number of remaining pairs is $2^{133.1}$ at the end of this phase.

In the process of *key recovery phase*, we need to guess 16 bytes of subkey RK_9, 4 bytes of subkey RK_8^* and 4 bytes of RK_0 to check if the 6-round of impossible differential is satisfied. While guessing the 4 bytes of RK_8^*, the probability that a pair can pass the test is about $Pr_2 = 2^{-8}$. The rest of the steps are similar to Subsection 3.2. At the end of this phase, there exist about $2^{133.1-96-22} = 2^{15.1}$ pairs.

After analyzing the remaining $2^{15.1}$ pairs, we can get rid of $2^{192} \cdot (1 - 2^{-8})^{2^{15.1}} \approx 2^{-6.3}$ wrong pairs. With about $2^{185.7}$ encryption trails the key can be recovered. The data complexity of this attack is about $2^{198.1}$ CP, the time complexity is about $2^{198.4}/9 \approx 2^{195.2}$ encryptions, and the memory we need for storing pairs is about $2^{133.1} \cdot 5 \cdot 32 = 2^{140.4}$ bytes.

As mentioned above, we can also guess three bytes of subkey RK_8^*, given 2^{208} CP, a 9-round attack can be mounted with the time complexity and memory about 2^{162} encryptions and 2^{117} bytes respectively. Moreover in the case that two bytes of RK_8^* are guessed, the data, time and memory complexity can be 2^{216} CP, 2^{130} encryptions and $2^{93.6}$ bytes respectively.

5 Conclusion

More powerful 6-round impossible differentials for both Rijndael-224 and Rijndael-256 are presented in this paper. Based on those, we significantly improve impossible differential attacks on both Rijndael-224 and Rijndael-256. The improvement can also result in a 10-round attack on Rijndael-256.

Acknowledgments. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

References

1. Bahrak, B., Aref, M.R.: A Novel Impossible Differential Cryptanalysis of AES. In: proceedings of WEWoRC (2007)
2. Barreto, P.S.L.M., Nikov, V., Nikova, S., Rijmen, V., Tischhauser, E.: Whirlwind: a new cryptographic hash function. Des. Codes Cryptography 56(2-3), 141–162 (2010)
3. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In: Stern, J. (ed.) Advances in Cryptology - EUROCRYPT '99. LNCS, vol. 1592, pp. 12–23. Springer (1999)
4. Biham, E., Dunkelman, O., Keller, N.: Related-Key Impossible Differential Attacks on 8-Round AES-192. In: Pointcheval, D. (ed.) CT-RSA. Lecture Notes in Computer Science, vol. 3860, pp. 21–33. Springer (2006)
5. Biryukov, A.: The Boomerang Attack on 5 and 6-Round Reduced AES. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) AES Conference. Lecture Notes in Computer Science, vol. 3373, pp. 11–15. Springer (2004)
6. Biryukov, A., Khovratovich, D.: Related-Key Cryptanalysis of the Full AES-192 and AES-256. In: Matsui, M. (ed.) ASIACRYPT. Lecture Notes in Computer Science, vol. 5912, pp. 1–18. Springer (2009)
7. Biryukov, A., Khovratovich, D., Nikolic, I.: Distinguisher and Related-Key Attack on the Full AES-256. In: Halevi, S. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 5677, pp. 231–249. Springer (2009)
8. Borst, J., Knudsen, L.R., Rijmen, V.: Two attacks on reduced idea. In: Fumy, W. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 1233, pp. 1–13. Springer (1997)
9. Daemen, J., Knudsen, L.R., Rijmen, V.: The Block Cipher Square. In: Biham, E. (ed.) FSE. Lecture Notes in Computer Science, vol. 1267, pp. 149–165. Springer (1997)
10. Daemen, J., Rijmen, V.: AES Proposal: Rijndael. In: 1st AES Conference, California, USA (1998)
11. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer (2002)
12. Demirci, H., Selçuk, A.A.: A Meet-in-the-Middle Attack on 8-Round AES. In: Nyberg, K. (ed.) FSE. Lecture Notes in Computer Science, vol. 5086, pp. 116–126. Springer (2008)
13. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., Whiting, D.: Improved Cryptanalysis of Rijndael. In: Schneier, B. (ed.) FSE. Lecture Notes in Computer Science, vol. 1978, pp. 213–230. Springer (2000)
14. FIPS 197: Advanced Encryption Standard. Federal Information Processing Standards Publication 197, U.S. Department of Commerce/N.I.S.T (2001)
15. Galice, S., Minier, M.: Improving Integral Attacks Against Rijndael-256 Up to 9 Rounds. In: Vaudenay, S. (ed.) AFRICACRYPT. Lecture Notes in Computer Science, vol. 5023, pp. 1–15. Springer (2008)
16. Gauravaram, P., Knudsen, L.R., Matusiewicz, K., Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: Grøstl - a SHA-3 candidate. Submission to NIST (2008), `http://www.groestl.info`
17. Gilbert, H., Minier, M.: A Collision Attack on 7 Rounds of Rijndael. In: AES Candidate Conference. pp. 230–241 (2000)
18. Jr., J.N., de Freitas, D.S., Phan, R.C.W.: New Multiset Attacks on Rijndael with Large Blocks. In: Dawson, E., Vaudenay, S. (eds.) Mycrypt. Lecture Notes in Computer Science, vol. 3715, pp. 277–295. Springer (2005)

19. Jr., J.N., Pavão, I.C.: Impossible-Differential Attacks on Large-Block Rijndael. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC. Lecture Notes in Computer Science, vol. 4779, pp. 104–117. Springer (2007)
20. Kim, J., Hong, S., Preneel, B.: Related-Key Rectangle Attacks on Reduced AES-192 and AES-256. In: Biryukov, A. (ed.) FSE. Lecture Notes in Computer Science, vol. 4593, pp. 225–241. Springer (2007)
21. Li, Y., Wu, W.: Improved Integral Attacks on Rijndael. Journal of Information Science and Engineering 27(6), 2031–2045 (2011)
22. Lu, J., Dunkelman, O., Keller, N., Kim, J.: New Impossible Differential Attacks on AES. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT. Lecture Notes in Computer Science, vol. 5365, pp. 279–293. Springer (2008)
23. Lucks, S.: Attacking Seven Rounds of Rijndael under 192-bit and 256-bit Keys. In: AES Candidate Conference. pp. 215–229 (2000)
24. Phan, R.C.W.: Impossible differential cryptanalysis of 7-round Advanced Encryption Standard (AES). Inf. Process. Lett. 91(1), 33–38 (2004)
25. Zhang, L., Wu, W., Park, J.H., Koo, B., Yeom, Y.: Improved Impossible Differential Attacks on Large-Block Rijndael. In: Wu, T.C., Lei, C.L., Rijmen, V., Lee, D.T. (eds.) ISC. Lecture Notes in Computer Science, vol. 5222, pp. 298–315. Springer (2008)
26. Zhang, W., Wu, W., Feng, D.: New Results on Impossible Differential Cryptanalysis of Reduced AES. In: Nam, K.H., Rhee, G. (eds.) ICISC. Lecture Notes in Computer Science, vol. 4817, pp. 239–250. Springer (2007)
27. Zhang, W., Wu, W., Zhang, L., Feng, D.: Improved Related-Key Impossible Differential Attacks on Reduced-Round AES-192. In: Biham, E., Youssef, A.M. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 4356, pp. 15–27. Springer (2006)
28. Zhang, W., Zhang, L., Wu, W., Feng, D.: Related-Key Differential-Linear Attacks on Reduced AES-192. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT. Lecture Notes in Computer Science, vol. 4859, pp. 73–85. Springer (2007)

A New 6-Round Impossible Differential of Rijndael-224 and 9-Round Attack with lower Data Complexity

Assume we start with round 1 and there is only one nonzero byte of the input difference ΔX_1 whereas the other bytes are zero. One options is depicted in Figure 5 with nonzero byte at the first byte position. Then encrypt the input for 2.5 rounds from the `SB` operation in 1R to the `SR` operation in 3R to get the difference ΔX_3^{SR}. Given the output difference with three nonzero bytes in the first column, whereas the other bytes are zero. For Rijndael-224, the only option exists is given in Figure 5. After 3.5 rounds decryption (as depicted from the operation `ARK`* in round 6 to the operation `ARK` in round 3 in order to get the difference ΔX_3^{MC}). For the first column of the state X_3^{SR}, the number of nonzero bytes of ΔX_3^{SR} is one, while the maximum number of nonzero bytes of ΔX_3^{MC} is three. Since the branch number of the MDS matrix is five, there exists an contradiction. Therefore, we make up a 6-round impossible differential for Rijndael-224.

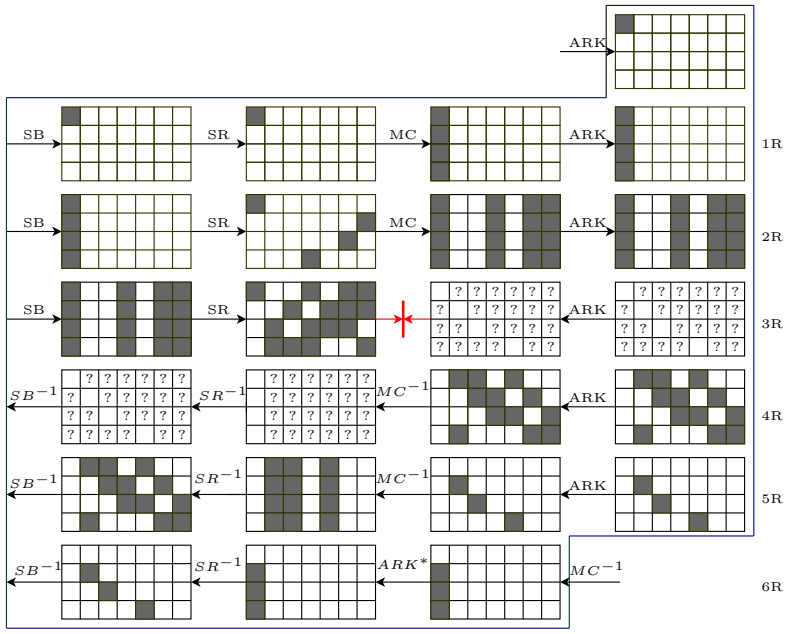


Fig. 5. The New 6-Round Impossible Differential of Rijndael-224

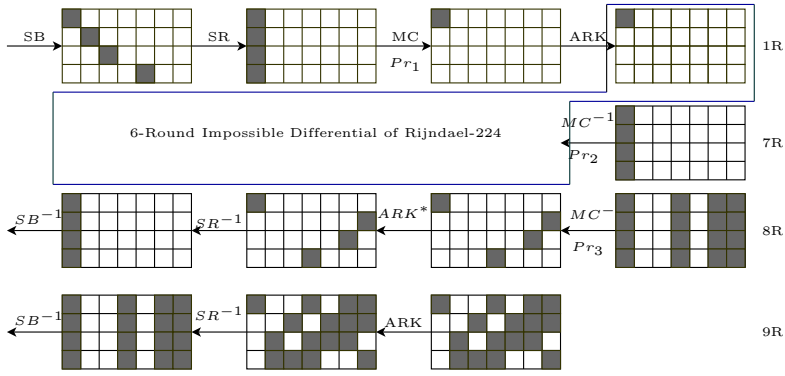


Fig. 6. The Improved 9-Round Attack on Rijndael-224 with lower Data Complexity

Chapter 11

Cryptanalysis of Reduced-Round SIMON32 and SIMON48

Publication Data

Q. Wang, Z. Liu, K. Varıcı, Y. Sasaki, V. Rijmen, Y. Todo: Cryptanalysis of Reduced-Round SIMON32 and SIMON48. In W. Meier and D. Mukhopadhyay (Eds.): *INDOCRYPT* 2014, volume 8885 of *Lecture Notes in Computer Science*, pages 143–160, 2014.

Contributions

Major author except Subsection 3.2.

Cryptanalysis of Reduced-round SIMON32 and SIMON48*

Qingju Wang[1,2], Zhiqiang Liu[1,2**], Kerem Varıcı[2,3**], Yu Sasaki[4**], Vincent Rijmen[2**], and Yosuke Todo[4**]

[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University, China
[2] KU Leuven, ESAT/COSIC and iMinds, Belgium
[3] ICTEAM-Crypto Group, Universite catholique de Louvain, Belgium
[4] NTT Secure Platform Laboratories, Japan

Abstract. SIMON family is one of the recent lightweight block cipher designs introduced by NSA. So far there have been several cryptanalytic results on this cipher by means of differential, linear and impossible differential cryptanalysis. In this paper, we study the security of SIMON32, SIMON48/72 and SIMON48/96 by using integral, zero-correlation linear and impossible differential cryptanalysis. Firstly, we present a novel experimental approach to construct the best known integral distinguishers of SIMON32. The small block size, 32 bits, of SIMON32 enables us to experimentally find a 15-round integral distinguisher, based on which we present a key recovery attack on 21-round SIMON32, while previous best results only achieved 19 rounds. Moreover, we attack 20-round SIMON32, 20-round SIMON48/72 and 21-round SIMON48/96 based on 11 and 12-round zero-correlation linear hulls of SIMON32 and SIMON48 respectively. Finally, we propose new impossible differential attacks which improve the previous impossible differential attacks. Our analysis shows that SIMON maintains enough security margin.
Keywords: SIMON, integral, zero-correlation, impossible differential

1 Introduction

Lightweight primitives are designed to be efficient for limited resource environments, but they should also ensure that the message is transmitted confidentially. Therefore, the vital design motivation is to maintain a reasonable trade-off between the security and performance. During recent years, many lightweight ciphers have been designed. Prominent examples are included but not limited to these: ICEBERG [2], mCrypton [3], HIGHT [4], PRESENT [5], KATAN [6], LED [7], Piccolo [8], KLEIN [9], EPCBC [10], PRINCE [11] and TWINE [12].

In 2013, NSA also proposed two families of highly-optimized block ciphers, SIMON and SPECK [13], which are flexible to provide excellent performance

* Due to page limitations, several details are omitted in this proceedings version. In particular, impossible differential attacks are only described in the full version [1].
** Corresponding authors.

in both hardware and software environments. Moreover both families offer large variety of block and key sizes such that the users can easily match the security requirements of their applications without sacrificing the performance. However, no cryptanalysis results are included in the specification of these algorithms.

Related Work and Our Contributions. On the one hand, several external cryptanalysis results on SIMON and SPECK were published. In [14, 15], differential attacks are presented on various state sizes of SIMON and SPECK, while the best linear attacks on SIMON are given in [16]. In [17] Biryukov et al. exploit the threshold search technique [18], where they showed better differential characteristics and proposed attacks with better results on several versions of SIMON and SPECK. Very recently, there are some differential attack results about SIMON32 and SIMON48 in ePrint [19]. These results need to be further verified although they seem intriguing.

In this paper, we investigate the security of SIMON32, SIMON48/72 and SIMON48/96 by using integral, zero-correlation linear and impossible differential cryptanalysis. We firstly apply integral cryptanalysis. Regarding SIMON32, because the block size is only 32 bits, we can experimentally observe the behaviors of all the plaintexts under a fixed key. Our experiments show that the number of distinguished rounds rapidly increases when the number of active bits becomes close to the block size. On the contrary, exploiting integral distinguishers with a large number of active bits for recovering the key is hard in general. Indeed, our distinguisher needs 31 active bits. To make the data complexity smaller than the code book, we cannot iterate the analysis even for two sets of the distinguisher. We then exploit the fact that the key schedule consists of simple linear equations, and show that reducing any fraction of subkey space can immediately reduce the main key space by solving the linear equations with Gaussian elimination. By combining several known cryptanalytic techniques we present an attack on 21-round SIMON32/64. As for SIMON48, the approach cannot be applied due to the large search space. However, according to the experimental results for SIMON32, we may expect that there exist good integral distinguishers of SIMON48 when the number of active bits is near the block size.

Moreover, we construct 11 and 12-round zero-correlation linear hulls of SIMON32 and SIMON48 respectively by using miss-in-the-middle technique. Then based on these distinguishers, we mount attacks on 20-round SIMON32, 20-round SIMON48/72 and 21-round SIMON48/96 delicately with the help of divide-and-conquer technique. Finally, we demonstrate impossible differential attacks on 18-round SIMON32, 18-round SIMON48/72 and 19-round SIMON48/96. Although these results are not better than the ones achieved by using differential, integral and zero-correlation linear cryptanalysis, they are the currently best impossible differential attacks for SIMON32 and SIMON48. Our improvements upon the state-of-the-art cryptanalysis for SIMON are given in Table 1.

Organization. The remainder of this paper is organized as follows. In Section 2, we give a brief description of SIMON. Section 3 covers the integral attack. In

Table 1. Summary of Attack Results on SIMON

Cipher	Full Rounds	Attack	Attacked Rounds	Complexity Time(EN)	Data	Memory(Bytes)	Source
SIMON32/64	32	Imp. Diff.	13	$2^{50.1}$	$2^{30.0}$KP	$2^{20.0}$	[20]
		Imp. Diff.	**18**	$2^{61.14}$	2^{32}KP	$2^{47.67}$	[1]
		Diff.	16	$2^{26.481}$	$2^{29.481}$CP	2^{16}	[15]
		Diff.	18	$2^{46.0}$	$2^{31.2}$CP	$2^{15.0}$	[14]
		Diff.	19	2^{32}	2^{31}CP	-	[17]
		Zero-Corr.	**20**	$2^{56.96}$	2^{32}KP	$2^{41.42}$	**Subsec 4.2**
		Integral	**21**	$2^{63.00}$	2^{31}CP	2^{54}	**Subsec 3.2**
SIMON48/72	36	**Imp. Diff.**	**18**	$2^{61.87}$	2^{48}KP	$2^{42.12}$	[1]
		Diff.	18	$2^{43.253}$	$2^{46.426}$CP	2^{24}	[15]
		Diff.	19	$2^{52.0}$	$2^{46.0}$CC	$2^{20.0}$	[14]
		Diff.	20	2^{52}	2^{46}CP	-	[17]
		Zero-Corr.	**20**	$2^{59.7}$	2^{48}KP	2^{43}	**Subsec 4.3**
SIMON48/96	36	Imp. Diff.	15	$2^{53.0}$	$2^{38.0}$KP	$2^{20.6}$	[20]
		Imp. Diff.	**19**	$2^{85.82}$	2^{48}KP	$2^{66.68}$	[1]
		Diff.	18	$2^{69.079}$	$2^{50.262}$CP	$2^{45.618}$	[15]
		Diff.	19	$2^{76.0}$	$2^{46.0}$CC	$2^{20.0}$	[14]
		Diff.	20	2^{75}	2^{46}CP	-	[17]
		Zero-Corr.	**21**	$2^{72.63}$	2^{48}KP	$2^{46.73}$	**Subsec 4.3**

CP: Chosen Plaintext; KP: Known Plaintext; CC: Chosen Ciphertext; EN: Encryption

Section 4, zero-correlation cryptanalysis is studied. Finally, we conclude the paper in Section 5. Impossible differential attacks are shown in [1]. Table 2 contains the notations that we use throughout this paper.

2 Brief Description of SIMON

We denote the SIMON block cipher using n-bit words by SIMON2n, with $n \in \{16, 24, 32, 48, 64\}$. SIMON2$n$ with an m-word key is referred to SIMON2n/mn.

SIMON is a two-branch balanced Feistel network with simple round functions consisting of three operations: AND (&), XOR ($\oplus$) and rotation ($\lll$). In round $i-1$, by using a function $F(x) = (x \lll 1)\&(x \lll 8) \oplus (x \lll 2)$, (L_{i-1}, R_{i-1}) are updated to (L_i, R_i) by $L_i = F(L_{i-1}) \oplus R_{i-1} \oplus k_{i-1}$ and $R_i = L_{i-1}$. The output of the last round (L_r, R_r) (r is the number of rounds) yields the ciphertext. The structure of the round function of SIMON is depicted in Figure 6 in Appendix A.

The key schedule of SIMON processes three different procedures depending on the key size. The first mn round keys are directly initialized with the main key, while the remaining key words are generated by three slightly different procedures depending on the key words value m:

$$k_{i+m} = c \oplus (z_j)_i \oplus k_i \oplus Y_m \oplus (Y_m \lll 1), \quad Y_m = \begin{cases} k_{i+1} \lll 3, & \text{if } m = 2, \\ k_{i+2} \lll 3, & \text{if } m = 3, \\ k_{i+3} \lll 3 \oplus k_{i+1}, & \text{if } m = 4. \end{cases}$$

Table 2. Notations: Top 8 are for general and bottom 4 are for integral attack.

L_r, R_r	left and right branches of the input state to the r-th round
$L_{r,\{i\sim j\}}, R_{r,\{i\sim j\}}$	the bits from bit i to bit j of L_r and R_r
$\Delta L_r, \Delta R_r$	left and right branches of the input difference of state to the r-th round
$\Gamma L_r, \Gamma R_r$	left and right branches of the input linear mask of state to the r-th round
$\Delta F(\cdot)$	the output difference after round function F
k_r	the subkey in the r-th round
$k_{r,\{i\sim j\}}$	the bits from bit i to bit j of k_r
?	an undetermined difference or linear mask
Let Λ be a collection of state vectors $X = (x_0, \ldots, x_{n-1})$ where $x_i \in \mathbb{F}_2$ is the i-th word of X:	
A	if all i-th words x_i in Λ are distinct, x_i is called active
B	if the sum of all i-th words x_i in Λ can be predicted, x_i is called balanced
C	if the values of all i-th words x_i in Λ are equal, x_i is called passive/constant
*	if the sum of all i-th words x_i in Λ can not be predicted

Here, the value c is constant `0xff...fc`, and $(z_j)_i$ denotes the i-th (least significant) bit from one of the five constant sequences z_j $(0 \le j \le 4)$. The main key can be derived if any sequence of m consecutive subkeys are known.

3 Integral Cryptanalysis of SIMON

The integral attack [21, 22] first constructs an integral distinguisher, which is a set of plaintexts such that the states after several rounds have a certain property, e.g. the XOR sum of all states in the set is 0. Then, several rounds are appended to the distinguisher for recovering subkeys. In this section, we investigate the integral properties and present integral attacks on 21-round SIMON32/64.

3.1 Integral Distinguishers of SIMON32

We experimentally find integrals of SIMON32. The results are shown in Table 3. Here the active bits are the ones in the input of round 1. An interesting observation is that the number of rounds increases rapidly when the number of active bits becomes close to the block size. Giving a theoretical reasoning for this observation seems hard. In other words, experimental approaches are useful for a small block size such that all plaintexts can be processed in a practical time.

Table 3. The Number of Rounds of SIMON32 Integral Distinguishers

Num. of Active Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Num. of Rounds	9	9	9	9	10	10	10	10	11	11	11	12	13	13	14	15

We explain the algorithm of our experiments as follows:

1. Firstly, we generate 2^t plaintexts ($t \geq 16$) by setting the right half (16 bits) and $(t-16)$ bits of the left half of the input in round 1 to be active, while keeping the remaining bits as constant.
2. (a) Choose the main key randomly. Encrypt 2^t plaintexts r rounds and check whether certain bits of the output are balanced (i.e., for each of these bits, the XOR sum of the bit over 2^t output states is 0). If yes, keep this as an integral candidate.
   (b) Repeat (a) 2^{13} times and verify if the integral candidate always holds. If not, discard it.
3. If there is an integral candidate for all the structures with the same pattern (i.e., with the same t active bits), we regard this as an r-round integral distinguisher of SIMON32.

As a result, we obtain a 15-round distinguisher (Figure 1) with 31 active bits:

$$\begin{aligned}(CAAA, AAAA, AAAA, AAAA,\ \ AAAA, AAAA, AAAA, AAAA)\\ \rightarrow (* * **, * * **, * * **, * * **,\ \ *B * *, * * **, B * **, * * *B).\end{aligned} \tag{1}$$

The distinguisher in (1) is not ensured for all of 2^{64} keys. Because our experiment did not return any failure, we expect that the success probability of this distinguisher is at least $1 - 2^{-13}$.

3.2 21-round Integral Attack of SIMON32/64

We use a 15-round integral distinguisher shown in Figure 1. We first prepare 2^{31} internal state values $(X_L \| X_R)$ in which 31 bits are active, then compute the corresponding plaintext $(L_0 \| R_0)$ as $L_0 \leftarrow X_R$ and $R_0 \leftarrow F(X_R) \oplus X_L$. Those 2^{31} plaintexts yield balanced bits in 3 positions after 15 rounds, i.e. (L_{15}, R_{15}). Moreover, the subsequent subkey XOR to R_{15} in round 16 never breaks the balanced property as long as the number of plaintexts in a set is even. We then mount a key recovery attack on 21-round SIMON-32/64 by adding six rounds after the distinguisher, which is illustrated in Figure 2.

3.2.1 Overall Strategy. The attacker guesses a part of the last 5-round subkeys $k_{16}, k_{17}, \ldots, k_{20}$. Then he partially decrypts the 2^{31} ciphertexts up to the state $R_{15} \oplus k_{15}$, and computes their XOR sum at the balanced bits. The 15-round distinguisher in Figure 1 has 3 balanced bits. Because the partial decryption up to all of those 3 bits requires too much subkey guesses, we only use 1 balanced bit at position 0. Thus, the subkey space can be reduced by 1 bit per set of 2^{31} plaintexts. In Figure 2, bit-position 0 of $(R_{15} \oplus k_{15})$ is circled and the related bits to the partial decryption are shown. 3 bits of k_{16}, 6 bits of k_{17}, 10 bits of k_{18}, 14 bits of k_{19}, 16 bits of k_{20}, in total 49 subkey bits are related. Because the block size is 32 bits, the analysis with 2^{31} plaintexts can be iterated at most twice, which implies that the 49-bit subkey space can be reduced at most 2 bits.

To detect the correct key, we further utilize the key schedule. 4 consecutive subkey values can reveal the main key value. We aim to recover 64 bits

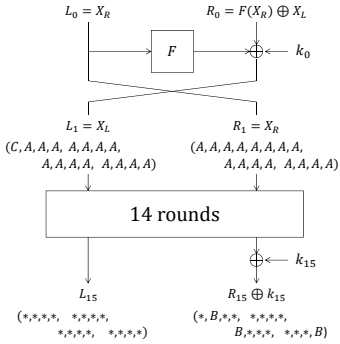


Fig. 1. 15-round Integral Distinguisher

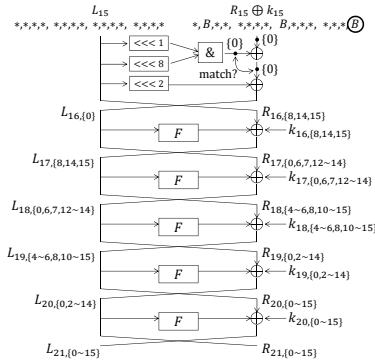


Fig. 2. 6-round Key-Recovery

of $k_{17}, \ldots, k_{20}$. Among 64 bits, 46 bits are suggested from the 6-round partial decryption. Moreover, because 5 subkeys $k_{16}, \ldots, k_{20}$ are linked only with linear equations, 3 bits of $k_{16,\{8,14,15\}}$ can be converted to 3-bit information for the remaining 18 bits of $k_{17}, \ldots, k_{20}$ by solving linear equations with Gaussian elimination. Thus, for each of 49 subkey bits suggested by the 6-round partial decryption, the attacker can obtain 64 bits of $k_{17}, \ldots, k_{20}$ only by guessing 15-bit information of $k_{17}, \ldots, k_{20}$, which leads to a faster key recovery attack than the exhaustive search.

3.2.2 Efficient Subkey Recovery. To perform the 6-round partial decryption with 49-bit subkey guess with a straight-forward method, partial decryption for 2^{31} ciphertexts with 2^{49} guesses are performed, which requires 2^{80} computations i.e. more than the exhaustive search. Several methods are known to reduce the complexity. Here, we use partial-sum [23], meet-in-the-middle match [24], and exploiting linearity for meet-in-the-middle match [25].

The attack finds 49 subkey bits satisfying $\bigoplus(R_{15} \oplus k_{15})_{\{0\}} = 0$, which is $\bigoplus((L_{15,\{15\}} \& L_{15,\{8\}}) \oplus L_{15,\{14\}} \oplus L_{16,\{0\}}) = 0$. This is further converted to

$$\bigoplus(L_{15,\{15\}} \& L_{15,\{8\}}) = \bigoplus(L_{15,\{14\}} \oplus L_{16,\{0\}}). \tag{2}$$

Hence, we can compute the left-hand side and right-hand side of Equation (2) independently, and later find the match between two independent computations as the meet-in-the-middle attack. The computation of the left-hand and right-hand side of Equation (2) is shown in the left and right part of Figure 3, in which 42 bits of subkeys are involved respectively. Compared to the original 6-round partial decryption in Figure 2, the number of related subkey bits are reduced from 49 to 42, which contributes to reduce the attack complexity. The complexity is further reduced by the partial-sum technique. Namely, every time subkey bits are guessed and state values are updated, we compress the amount of data only by keeping the state values appearing odd times.

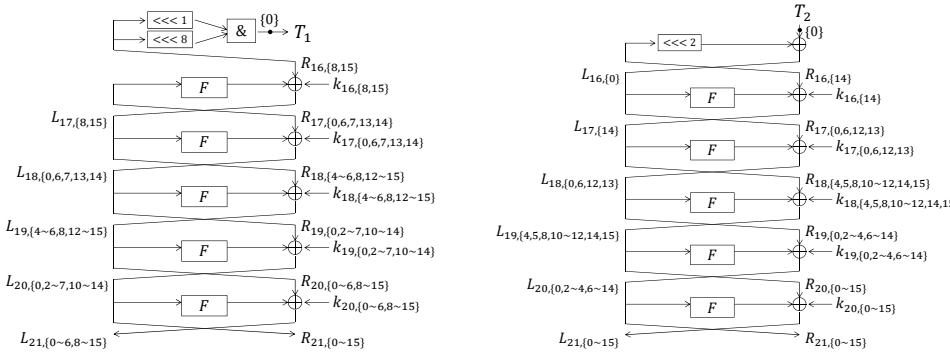


Fig. 3. Computations of $\bigoplus(L_{15,\{15\}} \& L_{15,\{8\}})$ and $\bigoplus(L_{15,\{14\}} \oplus L_{16,\{0\}})$

3.2.2.1 Computation of $\bigoplus(L_{15,\{15\}} \& L_{15,\{8\}})$. Given a set including 2^{31} plaintexts, $\bigoplus(L_{15,\{15\}} \& L_{15,\{8\}})$ for 2^{42} distinct subkey values can be computed with $2^{50.55}$ 21-round SIMON32 computations. The computed results along with 42-bit guessed subkeys are stored in a table T_1. We first initialize the following counters which remembers the parity of internal state values.

- 2^{27} counters T_{20}^x, each corresponding to $x = (L_{20,\{0,2\sim7,10\sim14\}}, R_{20,\{0\sim6,8\sim15\}})$.
- 2^{20} counters T_{19}^x, each corresponding to $x = (L_{19,\{4\sim6,8,12\sim15\}}, R_{19,\{0,2\sim7,10\sim14\}})$.
- 2^{13} counters T_{18}^x, each corresponding to $x = (L_{18,\{0,6,7,13,14\}}, R_{18,\{4\sim6,8,12\sim15\}})$.
- 2^7 counters T_{17}^x, each corresponding to $x = (L_{17,\{8,15\}}, R_{17,\{0,6,7,13,14\}})$.

We then compute $\bigoplus(L_{15,\{15\}} \& L_{15,\{8\}})$ by the following procedure.

1. For 2^{15} guesses of $k_{20,\{0\sim6,8\sim15\}}$ and for each 2^{31} ciphertext values, calculate 27 bits of $(L_{20,\{0,2\sim7,10\sim14\}}, R_{20,\{0\sim6,8\sim15\}})$, and increase the relevant counter T_{20}^x by 1. Keep the values of $(L_{20,\{0,2\sim7,10\sim14\}}, R_{20,\{0\sim6,8\sim15\}})$ which appear odd times.
2. For 2^{12} guesses of $k_{19,\{0,2\sim7,10\sim14\}}$ and for each 2^{27} remaining values, calculate 20 bits of $(L_{19,\{4\sim6,8,12\sim15\}}, R_{19,\{0,2\sim7,10\sim14}$) and increase the counter T_{19}^x. Keep the values which appear odd times.
3. For 2^8 guesses of $k_{18,\{4\sim6,8,12\sim15\}}$ and for each 2^{20} remaining values, calculate 13 bits of $(L_{18,\{0,6,7,13,14\}}, R_{18,\{4\sim6,8,12\sim15\}})$ and increase the counter T_{18}^x. Keep the values which appear odd times.
4. For 2^5 guesses of $k_{17,\{0,6,7,13,14\}}$ and for each 2^{13} remaining values, calculate 7 bits of $(L_{17,\{8,15\}}, R_{17,\{0,6,7,13,14\}})$ and increase the counter T_{17}^x. Keep the values which appear odd times.
5. For 2^2 guesses of $k_{16,\{8,15\}}$ and for each 2^7 remaining values, calculate 2 bits of $L_{15,\{8,15\}}$ and then 1-bit of $(L_{15,\{15\}} \& L_{15,\{8\}})$. Store it in a table T_1 along with the guesses for 42-bit subkeys.

We then evaluate the computational cost. The unit is a single execution of 21-round SIMON32. Updating one bit of the state is equivalent to $1/(16 \cdot 21)$ 21-round SIMON32 computation.

Step 1. $2^{31} \cdot 2^{15} \cdot 15/(16 \cdot 21) \approx 2^{41.51}$.
Step 2. $2^{27} \cdot 2^{15} \cdot 2^{12} \cdot 12/(16 \cdot 21) \approx 2^{49.19}$.
Step 3. $2^{20} \cdot 2^{15} \cdot 2^{12} \cdot 2^{8} \cdot 8/(16 \cdot 21) \approx 2^{49.61}$.
Step 4. $2^{13} \cdot 2^{15} \cdot 2^{12} \cdot 2^{8} \cdot 2^{5} \cdot 5/(16 \cdot 21) \approx 2^{46.93}$.
Step 5. $2^{7} \cdot 2^{15} \cdot 2^{12} \cdot 2^{8} \cdot 2^{5} \cdot 2^{2} \cdot 3/(16 \cdot 21) \approx 2^{42.19}$.

The sum of the above 5 steps is $2^{50.55}$ 21-round SIMON32 computations. The table T_1 contains 2^{42} elements of 43-bit information, which is less than 2^{45} bytes.

3.2.2.2 Computation of $\bigoplus(L_{15,\{14\}} \oplus L_{16,\{0\}})$. For each of 2^{31} plaintexts set, $\bigoplus(L_{15,\{14\}} \oplus L_{16,\{0\}})$ for distinct 2^{42} subkey values can be computed with $2^{54.01}$ 21-round SIMON32 computations. The computed results along with 42-bit guessed subkeys are stored in a table T_2. Because the procedure is similar to the computation of T_1, the attack is explained shortly.

1. For 2^{16} guesses of $k_{20,\{0\sim15\}}$ and 2^{31} ciphertext values, calculate 29 bits of $(L_{20,\{0,2\sim4,6\sim14\}}, R_{20,\{0\sim15\}})$. The complexity of this step is $2^{31} \cdot 2^{16} \cdot 16/(16 \cdot 21) \approx 2^{42.61}$.
2. For 2^{13} guesses of $k_{19,\{0,2\sim4,6\sim14\}}$ and 2^{29} remaining values, calculate 21 bits of $(L_{19,\{4,5,8,10\sim12,14,15\}}, R_{19,\{0,2\sim4,6\sim14}})$. The complexity of this step is $2^{29} \cdot 2^{16} \cdot 2^{13} \cdot 13/(16 \cdot 21) \approx 2^{53.31}$.
3. For 2^{8} guesses of $k_{18,\{4,5,8,10\sim12,14,15\}}$ and 2^{21} remaining values, calculate 12 bits of $(L_{18,\{0,6,12,13\}}, R_{18,\{4,5,8,10\sim12,14,15\}})$. The complexity of this step is $2^{21} \cdot 2^{16} \cdot 2^{13} \cdot 2^{8} \cdot 8/(16 \cdot 21) \approx 2^{52.61}$.
4. For 2^{4} guesses of $k_{17,\{0,6,12,13\}}$ and 2^{12} remaining values, calculate 5 bits of $(L_{17,\{14\}}, R_{17,\{0,6,12,13\}})$. The complexity of this step is $2^{12} \cdot 2^{16} \cdot 2^{13} \cdot 2^{8} \cdot 2^{4} \cdot 4/(16 \cdot 21) \approx 2^{46.61}$.
5. For 2 guesses of $k_{16,\{14\}}$ and 2^{5} remaining values, calculate 2 bits of $L_{15,\{14\}}$ and then 1-bit of $(L_{15,\{14\}} \oplus L_{16,\{0\}})$. Store it in a table T_2 along with the guesses for 42-bit subkeys. The complexity of this step is $2^{5} \cdot 2^{16} \cdot 2^{13} \cdot 2^{8} \cdot 2^{4} \cdot 2 \cdot 2/(16 \cdot 21) \approx 2^{39.61}$.

Table T_2 contains 2^{42} elements of 43-bit information, which is less than 2^{45} bytes.

3.2.2.3 Matching T_1 *and* T_2. After T_1 and T_2 are independently generated, we derive valid 49-bit subkey candidates. Because both of T_1 and T_2 contain 2^{42} elements, the number of pairs is 2^{84}. From Equation (2), the valid candidates will match the 1-bit result in T_1 and T_2. Moreover, 42-bit subkeys used in T_1 and 42-bit subkeys in T_2 overlap in 35 bits. Thus, $2^{84-1-35} = 2^{48}$ valid candidates are generated, which reduces the entire 49-bit space by one bit.

3.2.3 Entire Attack Procedure and Complexity Evaluation

1. Represent the three subkey bits $k_{15,\{8,14\sim15\}}$ by using $k_{16}\|k_{17}\|k_{18}\|k_{19}$ according to the key schedule of SIMON32 and keep the three linear equations.
2. Generate a set of 2^{31} plaintexts.

3. For each of 2^{31} plaintexts, compute T_1 and T_2 as explained before, and identify the correct key candidates to reduce the subkey space of 49 bits in the last 6 rounds.
4. For each of remaining subkey candidates, guess the 15 bits $k_{19,\{1,15\}} \| k_{18,\{0\sim3,7,9\}} \| k_{17,\{1\sim5,11,15\}}$ and obtain three bits of $k_{17,\{8\sim10\}}$ by solving the linear equations with Gaussian elimination. Then compute all bits of the original key by inverting the key schedule, and check the correctness of the guess by using two plaintext-ciphertext pairs.

The data complexity of the attack is 2^{31} chosen-plaintexts. The time complexity for Step 3 is $2^{50.55} + 2^{54.01} \approx 2^{54.13}$ 21-round SIMON32 computations. After Step 3, 2^{48} subkey candidates remain. In Step 4, the cost of Gaussian elimination is much smaller than 21-round SIMON32, and thus is ignored. The check with two plaintext-ciphertext pairs can be done one by one, that is, the check for the second pair is performed only with the first check is passed with probability 2^{-32}. Hence, the time complexity is $2^{48} \cdot 2^{15}(1+2^{-32}) \approx 2^{63}$ 21-round SIMON32 computations. In total, the time complexity is $2^{54.13} + 2^{63} \approx 2^{63.00}$ 21-round SIMON32 computations. The memory complexity is $2 \cdot 2^{45}$ bytes for constructing T_1 and T_2 and 2^{48} 49-bit subkey candidates after analyzing a plaintext set, which is less than 2^{51} bytes. The success probability is $1 - 2^{-13}$ due to the probability of the 15-round distinguisher.

4 Zero-Correlation Linear Cryptanalysis of SIMON

The zero-correlation attack is one of the recent cryptanalytic method introduced by Bogdanov and Rijmen [26]. The attack is based on linear approximations with zero correlation (i.e. linear approximations with probability exactly 1/2). We introduce 11 and 12-round zero-correlation linear approximations of SIMON32 and SIMON48, based on which we present key recovery attacks on 20-round SIMON32, 20-round SIMON48/72 and 21-round SIMON48/96 respectively.

4.1 Zero-Correlation Linear Distinguishers of SIMON

By applying miss-in-the-middle technique, we construct 11-round zero-correlation linear hull for SIMON32 (see Figure 4). More specifically, this distinguisher consists of two parts: forward part (along the encryption direction) and backward part (along the decryption direction). For the forward part, we find that for any 6-round non-zero correlation linear hull with input mask being (`0x0001,0x0000`), the most significant bit of the left half of its output mask must be 0. As to the backward part, we observe that for any 5-round non-zero correlation linear hull with input mask being (`0x0000,0x0080`), the most significant bit of the left half of its output mask must be 1. Combining the above two parts, we can deduce that an 11-round linear hull with input and output masks being (`0x0001,0x0000`) and (`0x0000,0x0080`) must be a zero-correlation linear hull. Similarly, a 12-round zero-correlation linear hull for SIMON48 can be derived (see Table 4 in Appendix B).

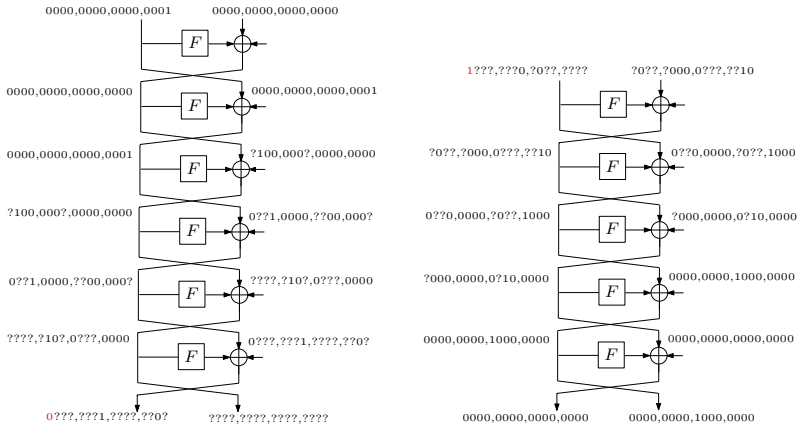


Fig. 4. Zero-Correlation Linear Approximations of 11-round SIMON32. The '0' at bottom left and the '1' at top right (in red) constitute the contradiction that ensures correlation zero.

4.2 Zero-Correlation Linear Attack on 20-round SIMON32

Let E denote the 20-round SIMON32 from round 0 to round 19. Suppose that the 11-round zero-correlation linear distinguisher given in Figure 4 covers from round 5 to round 15. We now present an attack on E based on this distinguisher by adding five rounds before the distinguisher and four rounds after the distinguisher, which is illustrated in Figure 5.

4.2.1 Overall Strategy. For each of the 2^{32} plaintext-ciphertext pairs, the attacker first guesses a part of the last 4-round subkeys $k_{16}, k_{17}, k_{18}, k_{19}$ and partially decrypts the ciphertext up to the state $R_{16,\{7\}}$. Then he guesses a part of the first 5-round subkeys $k_0, k_1, \ldots, k_4$ and partially encrypts the plaintext up to the state $L_{5,\{0\}}$. Finally, the attacker computes the value of $L_{5,\{0\}} \oplus R_{16,\{7\}}$. The subkey bits related to the above partial encryption and partial decryption are shown in Figure 5. We can see that 14 bits of k_0, 10 bits of k_1, 6 bits of k_2, 3 bits of k_3, one bit of k_4, one bit of k_{16}, 3 bits of k_{17}, 6 bits of k_{18}, 10 bits of k_{19}, in total 54 subkey bits are related.

For a guessed value of the 54 subkey bits, if the event that $L_{5,\{0\}} \oplus R_{16,\{7\}}$ is equal to 0 happens 2^{31} times (i.e., the correlation of the linear equation $L_{5,\{0\}} \oplus R_{16,\{7\}} = 0$ is exactly 0), then we take this guessed subkey information as a correct subkey candidate. According to [26] and the Wrong-Key Randomization Hypothesis given in [27], for a wrong subkey candidate, the probability that the correlation of $L_{5,\{0\}} \oplus R_{16,\{7\}} = 0$ is 0 can be estimated as $\frac{1}{\sqrt{2\pi}} 2^{\frac{4-32}{2}} \approx 2^{-15.33}$. Thus the 54-bit subkey space can be reduced by a factor of $2^{15.33}$ approximately.

In order to recover the master key value (i.e., k_0, k_1, k_2, k_3), we further exploit the key schedule. Among 64 bits of the master key, 33 bits are suggested from the

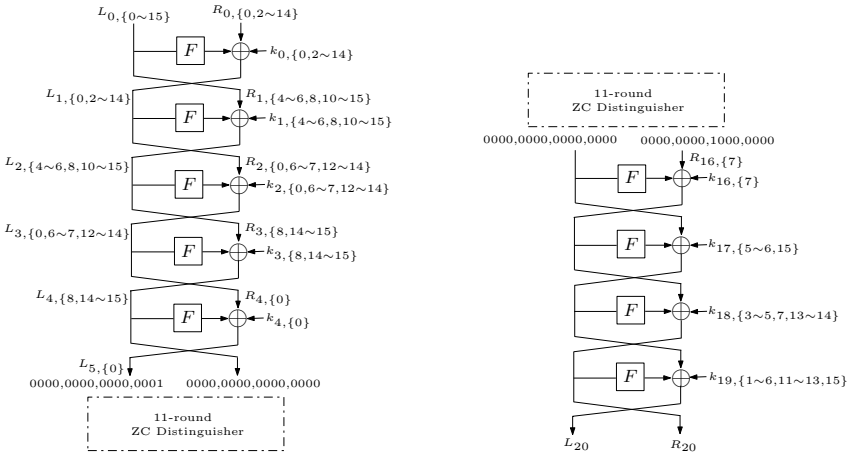


Fig. 5. Add 5 rounds before and 4 rounds after the Distinguisher

above procedure. Moreover, k_4, $k_{16}, k_{17}, k_{18}, k_{19}$ can be derived from the master key by using linear equations, therefore, one bit of k_4, one bit of k_{16}, 3 bits of k_{17}, 6 bits of k_{18} and 10 bits of k_{19} (totally 21 subkey bits) can be converted to 21-bit information for the remaining 31 bits of the master key. More specifically, for each of the 33 master key bits suggested above, the attacker can guess 10-bit information of the master key and then obtain 21 linear equations of 21 variables (i.e., the remaining 21 bits of the master key). By solving these linear equations with Gaussian elimination, the attacker can retrieve the master key value.

4.2.2 Efficient Subkey Recovery. We now explain the strategy for efficiently performing 4-round partial decryption and 5-round partial encryption with 54-bit subkey guess. By using a straightforward approach, we need to do the partial decryption and partial encryption for 2^{32} plaintext-ciphertext pairs with 2^{54} subkey guesses. This requires $2^{32+54} = 2^{86}$ computations, which is much more than the exhaustive key search. In our attack, we adopt the divide-and-conquer technique delicately to reduce the time complexity. More specifically, checking whether $L_{5,\{0\}} \oplus R_{16,\{7\}} = 0$ has a zero correlation or not can be done by counting the number of occurrences of the event that $L_{5,\{0\}} \| R_{16,\{7\}}$ is equal to "00" or "11" (If this number is 2^{31}, then the correlation of $L_{5,\{0\}} \oplus R_{16,\{7\}} = 0$ is exactly zero). To do this, we first guess the 20 bits of the last four-round subkeys relevant to $R_{16,\{7\}}$ and get the value of $L_{0,\{0\sim15\}} \| R_{0,\{0,2\sim14\}} \| R_{16,\{7\}}$ (regarded as the starting state), based on which, we set a starting counter and update the state bit-by-bit for the first six rounds (the counters corresponding to the states are obtained accordingly). Eventually we derive the counter with respect to the value of $L_{5,\{0\}} \| R_{16,\{7\}}$. Note that all the bit-by-bit state transitions are chosen elaborately to make the time complexity of our attack optimal, and all the counters involved in this attack need to be initialized firstly. The reason why

we do not use all the plaintext-ciphertext bits related to $L_{5,\{0\}}$ and $R_{16,\{7\}}$ as the starting state is that the size of this state is too large for us to mount an efficient attack. The detailed attack procedure is given as below.

1. Collect all the 2^{32} plaintext-ciphertext pairs of E. Let T_1 be a vector of 2^{31} counters correspond to all possible values of $L_{0,\{0\sim15\}}\|R_{0,\{0,2\sim14\}}\|R_{16,\{7\}}$ (denoted as S_1^1). Guess the 20 subkey bits $k_{16,\{7\}}\|k_{17,\{5\sim6,15\}}\|k_{18,\{3\sim5,7,13\sim14\}}\|k_{19,\{1\sim6,11\sim13,15\}}$. Then for each plaintext-ciphertext pair:
   (a) Do partial decryption to get the value of $R_{16,\{7\}}$ and increase the corresponding counter T_{1,S_1^1} by one according to the value of S_1^1. After that, we will do bit-by-bit state transitions based on S_1^1 and update the counters corresponding to the intermediate states.
   (b) Let T_2 be a vector of 2^{30} counters which correspond to all possible values of $L_{0,\{1\sim15\}}\|R_{0,\{0,3\sim7,9\sim14\}}\|L_{1,\{2,8\}}\|R_{16,\{7\}}$ (denoted as S_2^1). Guess the subkey bits $k_{0,\{2,8\}}$. Encrypt partially for each possible value of S_1^1 to obtain the value of $L_{1,\{2,8\}}$, then add T_{1,S_1^1} to the relevant counter T_{2,S_2^1} according to the value of S_2^1.
   (c) Guess subkey bits $k_{0,\{9\}}$, $k_{0,\{3\}}$, $k_{0,\{4,10\}}$, $k_{0,\{11\}}$, $k_{0,\{5\}}$ and $k_{0,\{0,6\sim7,12\sim14\}}$ step by step (see Table 5 in Appendix B).[1] Do similarly to the above and finally get the values of the counters corresponding to the state $L_{1,\{0,2\sim14\}}\|R_{1,\{4\sim6,8,10\sim15\}}\|R_{16,\{7\}}$ (denoted as S_0^2).
2. Let X_1 be a vector of 2^{24} counters which correspond to all possible values of $L_{1,\{0,2\sim7,9\sim14\}}\|R_{1,\{4\sim6,8,11\sim15\}}\|L_{2,\{10\}}\|R_{16,\{7\}}$ (denoted as S_1^2). Guess the subkey bit $k_{1,\{10\}}$. For each possible value of S_0^2, do partial encryption to derive the value of $L_{2,\{10\}}$ and add T_{8,S_0^2} to the corresponding counter X_{1,S_1^2} according to the value of S_1^2. After that, guess the subkey bits $k_{1,\{4\}}$, $k_{1,\{11\}}$, $k_{1,\{12\}}$, $k_{1,\{13\}}$, $k_{1,\{5\}}$, $k_{1,\{6\}}$ and $k_{1,\{8,14\sim15\}}$ sequentially. Do similarly to the above and eventually obtain the values of the counters corresponding to the state $L_{2,\{4\sim6,8,10\sim15\}}\|R_{2,\{0,6\sim7,12\sim14\}}\|R_{16,\{7\}}$ (denoted as S_0^3).
3. Let Y_1 be a vector of 2^{16} counters which correspond to all possible values of $L_{2,\{4\sim6,8,11\sim15\}}\|R_{2,\{0,6\sim7,13\sim14\}}\|L_{3,\{12\}}\|R_{16,\{7\}}$ (denoted as S_1^3). Guess the subkey bit $k_{2,\{12\}}$. For each possible value of S_0^3, do partial encryption to gain the value of $L_{3,\{12\}}$ and add X_{8,S_0^3} to the relevant counter Y_{1,S_1^3} according to the value of S_1^3. Then guess the subkey bits $k_{2,\{13\}}$, $k_{2,\{14\}}$, $k_{2,\{6\}}$, $k_{2,\{7\}}$ and $k_{2,\{0\}}$ step by step. Do similarly to the above and finally derive the values of the counters corresponding to the state $L_{3,\{0,6\sim7,12\sim14\}}\|R_{3,\{8,14\sim15\}}\|R_{16,\{7\}}$ (denoted as S_0^4).
4. Let Z_1 be a vector of 2^9 counters which correspond to all possible values of $L_{3,\{0,6\sim7,12\sim13\}}\|R_{3,\{8,15\}}\|L_{4,\{14\}}\|R_{16,\{7\}}$ (denoted as S_1^4). Guess the subkey bit $k_{3,\{14\}}$. For each possible value of S_0^4, do partial encryption to get the value of $L_{4,\{14\}}$ and add Y_{6,S_0^4} to the corresponding counter Z_{1,S_1^4} according to the value of S_1^4. After that, guess the subkey bits $k_{3,\{15\}}$ and $k_{3,\{8\}}$ step by step. Do similarly to the above and eventually get the values of the counters corresponding to the state $L_{4,\{8,14\sim15\}}\|R_{4,\{0\}}\|R_{16,\{7\}}$ (denoted as S_0^5).

[1] Please refer to the full version for more details of the subsequential attack procedures.

5. Let W be a vector of 2^2 counters which correspond to all possible values of $L_{5,\{0\}}\|R_{16,\{7\}}$. Guess the subkey bit $k_{4,\{0\}}$. For each possible value of S_0^5, do partial encryption to obtain the value of $L_{5,\{0\}}$ and add Z_{3,S_0^5} to the relevant counter in W according to the value of $L_{5,\{0\}}\|R_{16,\{7\}}$. If $W_0 + W_3 = 2^{31}$ (Note that W_0, W_3 are the counters corresponding to the cases that $L_{5,\{0\}}\|R_{16,\{7\}}$ = "00" and $L_{5,\{0\}}\|R_{16,\{7\}}$ = "11", respectively), keep the guessed 54-bit subkey information (i.e., $k_{0,\{0,2\sim14\}}\|k_{1,\{4\sim6,8,10\sim15\}}\|k_{2,\{0,6\sim7,12\sim14\}}\|k_{3,\{8,14\sim15\}}\|k_{4,\{0\}}\|k_{16,\{7\}}\|k_{17,\{5\sim6,15\}}\|k_{18,\{3\sim5,7,13\sim14\}}\|k_{19,\{1\sim6,11\sim13,15\}}$, denoted as η) as a possible subkey candidate, and discard it otherwise.

According to [26] and the Wrong-Key Randomization Hypothesis given in [27], the probability that a wrong subkey candidate for η is kept after Step 5 can be approximated as $\frac{1}{\sqrt{2\pi}}2^{-14} \approx 2^{-15.33}$, thus about $2^{54} \times 2^{-15.33} = 2^{38.67}$ subkey candidates for η will be left after the above procedure.

4.2.3 Master Key Recovery.

1. Represent the subkey bits $k_{4,\{0\}}$, $k_{16,\{7\}}$, $k_{17,\{5\sim6,15\}}$, $k_{18,\{3\sim5,7,13\sim14\}}$ and $k_{19,\{1\sim6,11\sim13,15\}}$ by using $k_{0,\{0\sim15\}}$, $k_{1,\{0\sim15\}}$, $k_{2,\{0\sim15\}}$ and $k_{3,\{0\sim15\}}$ according to the key schedule of SIMON32 and keep these 21 linear equations.
2. For each of the remaining $2^{38.67}$ values of η, do the following to recover the 64-bit master key:
   (a) Guess the 10 subkey bits $k_{0,\{1,15\}}$, $k_{1,\{0\sim3,7,9\}}$ and $k_{2,\{1\sim2\}}$ and obtain 21 linear equations with respect to $k_{2,\{3\sim5,8\sim11,15\}}$ and $k_{3,\{0\sim7,9\sim13\}}$.
   (b) Solve the linear equations by means of Gaussian elimination so as to get the value of $k_{2,\{3\sim5,8\sim11,15\}}\|k_{3,\{0\sim7,9\sim13\}}$, thus all bits of master key can be gained. Verify whether the master key is correct or not by using two plaintext-ciphertext pairs (do the verification for one pair firstly, if the master key can pass the test, do the verification for the other pair).

4.2.4 Complexity of the Attack. The data complexity of this attack is 2^{32} known plaintexts. The memory complexity is primarily owing to keeping the remaining subkey candidates for η in Step 5 of the *Efficient subkey recovery* phase, thus it can be estimated as $2^{38.67} \cdot 54/8 \approx 2^{41.42}$ bytes.

Regarding the time complexity of this attack, it is mainly dominated by Steps 1–4 of the *Efficient subkey recovery* phase and Step 2(b) of the *Master key recovery* phase, which can be derived as follows.

1. In Step 1 of the *Efficient subkey recovery* phase, the time complexity can be estimated as $2^{52}/5 + 3 \cdot 2^{48}/5 + 2 \cdot 2^{47}/5 + 2^{49}/5 + 2^{54} \cdot 3/5 \approx 2^{53.42}$ 20-round SIMON32 encryptions (See Table 5 in Appendix B).
2. In Step 2 of the *Efficient subkey recovery* phase, the time complexity can be estimated as $7 \cdot 2^{54}/5 + 2^{55} \cdot 3/5 \approx 2^{55.38}$ 20-round SIMON32 encryptions.
3. In Step 3 of the *Efficient subkey recovery* phase, the time complexity can be measured as $3 \cdot 2^{56}/5 + 2 \cdot 2^{55}/5 + 2^{54}/5 \approx 2^{55.77}$ 20-round SIMON32 encryptions.

4. In Step 4 of the *Efficient subkey recovery* phase, the time complexity can be measured as $2 \cdot 2^{55}/5 + 2^{54}/5 = 2^{54}$ 20-round SIMON32 encryptions.
5. In Step 2(b) of the *Master key recovery* phase, solving 21 linear equations with 21 variables by using Gaussian elimination needs about $\frac{1}{3} \cdot 21^3 \approx 3087$ bit-XOR operations, which can be measured by $\frac{3087}{16 \cdot 4 \cdot 20} \approx 2^{1.27}$ 20-round SIMON32 encryptions (Note that there are three XOR operations and one AND operation in the round function of SIMON. For simplicity, we approximate them as four XOR operations in our analysis), thus the time complexity of this step can be approximated as $2^{38.67} \cdot 2^{10} \cdot 2^{1.27} + 2^{38.67} \cdot 2^{10} \approx 2^{50.44}$ 20-round SIMON32 encryptions.

Therefore, the total time complexity of this attack is about $2^{53.42} + 2^{55.38} + 2^{55.77} + 2^{54} + 2^{50.44} \approx 2^{56.96}$ 20-round SIMON32 encryptions.

4.3 Zero-Correlation Linear Attacks on SIMON48

Similarly, by using the 12-round zero-correlation linear distinguisher in Table 4 in Appendix, we can mount key recovery attacks on 20-round SIMON48/72 and 21-round SIMON48/96. For the former, the data, memory and time complexities are about 2^{48} known plaintexts, 2^{43} bytes and $2^{59.7}$ 20-round SIMON48/72 encryptions, respectively. As to the latter, the data, memory and time complexities are about 2^{48} known plaintexts, $2^{46.73}$ bytes and $2^{72.63}$ 21-round SIMON48/96 encryptions, respectively.

5 Discussion and Conclusion

Discussion. As mentioned before, applying our experiments to SIMON48 is hard due to the large block size especially when the number of active bits is close to the block size. We then did experiments in which the number of active bits is 24 (i.e., half of the state) and 30 (i.e., 5/8 of the state), and found 9 and 10-round distinguishers, respectively. Interestingly, according to the experimental results for SIMON32 in Table 3, we observed that if half of the state (16 bits) are active, 9-round distinguishers can be found, and if 5/8 of the state (20 bits) are active, 10-round distinguishers can be derived. It seems that the ratio between the number of active bits and the block size for SIMON48 matches with SIMON32 well, thus we may find 13-round distinguisher with 7/8 of the state (42 bits) being active and 15-round distinguisher with 47 active bits for SIMON48. It remains an open problem to apply this experimental approach efficiently to block ciphers with larger block size.

Conclusion. In this paper, we investigated the security of SIMON32 and SIMON48 by using integral, zero-correlation linear and impossible differential cryptanalysis, and obtained some new results on these ciphers. Firstly, we introduced a novel approach to find a 15-round integral distinguisher of SIMON32,

with which an efficient attack was mounted on 21-round SIMON32. This approach gives a new way of constructing integral distinguishers for block ciphers with small block size. Secondly, we presented attacks on 20-round SIMON32, 20-round SIMON48/72 and 21-round SIMON48/96 delicately based on 11 and 12-round zero-correlation linear hulls of SIMON32 and SIMON48 respectively. Our attacks improved the previous best results (appeared in FSE 2014) in terms of the number of attacked rounds. Moreover, we proposed improved impossible differential attacks on SIMON32 and SIMON48. It is expected that our results could be beneficial to the security evaluation of SIMON.

Acknowledgments The authors are grateful to all anonymous reviewers for their valuable comments. We also thank Lauren De Meyer, Tomer Ashur and Andras Boho for helping with the integral distinguishers. Moreover, the authors are supported by the National Natural Science Foundation of China (no. 61202371), Major State Basic Research Development Program (973 Plan, no. 2013CB338004), China Postdoctoral Science Foundation (no. 2012M521829) and Shanghai Postdoctoral Research Funding Program (no. 12R21414500).

References

1. Qingju Wang, Zhiqiang Liu, Kerem Varıcı, Yu Sasaki, Vincent Rijmen, and Yosuke Todo. Cryptanalysis of Reduced-round SIMON32 and SIMON48. Cryptology ePrint Archive, 2014. `http://eprint.iacr.org/`.
2. François-Xavier Standaert, Gilles Piret, Gaël Rouvroy, Jean-Jacques Quisquater, and Jean-Didier Legat. ICEBERG : An Involutional Cipher Efficient for Block Encryption in Reconfigurable Hardware. In Bimal K. Roy and Willi Meier, editors, *FSE*, volume 3017 of *Lecture Notes in Computer Science*, pages 279–299. Springer, 2004.
3. Chae Hoon Lim and Tymur Korkishko. mCrypton - A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors. In JooSeok Song, Taekyoung Kwon, and Moti Yung, editors, *WISA*, volume 3786 of *Lecture Notes in Computer Science*, pages 243–258. Springer, 2005.
4. Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bonseok Koo, Changhoon Lee, Donghoon Chang, Jaesang Lee, Kitae Jeong, Hyun Kim, Jongsung Kim, and Seongtaek Chee. HIGHT: A New Block Cipher Suitable for Low-Resource Device. In Louis Goubin and Mitsuru Matsui, editors, *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pages 46–59. Springer, 2006.
5. 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.
6. Christophe De Cannière, Orr Dunkelman, and Miroslav Knezevic. KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers. In Christophe Clavier and Kris Gaj, editors, *CHES*, volume 5747 of *LNCS*, pages 272–288. Springer, 2009.

7. Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED Block Cipher. In Preneel and Takagi [28], pages 326–341.
8. Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita, and Taizo Shirai. Piccolo: An Ultra-Lightweight Blockcipher. In Preneel and Takagi [28], pages 342–357.
9. Zheng Gong, Svetla Nikova, and Yee Wei Law. KLEIN: A New Family of Lightweight Block Ciphers. In Ari Juels and Christof Paar, editors, *RFIDSec*, volume 7055 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2011.
10. Huihui Yap, Khoongming Khoo, Axel Poschmann, and Matt Henricksen. EPCBC - A Block Cipher Suitable for Electronic Product Code Encryption. In Dongdai Lin, Gene Tsudik, and Xiaoyun Wang, editors, *CANS*, volume 7092 of *Lecture Notes in Computer Science*, pages 76–97. Springer, 2011.
11. Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2012.
12. Tomoyasu Suzaki, Kazuhiko Minematsu, Sumio Morioka, and Eita Kobayashi. TWINE: A Lightweight Block Cipher for Multiple Platforms. In Knudsen and Wu [29], pages 339–354.
13. Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404, 2013.
14. Farzaneh Abed, Eik List, Jakob Wenzel, and Stefan Lucks. Differential Cryptanalysis of round-reduced Simon and Speck. In Carlos Cid and Christian Rechberger, editors, *International Workshop on Fast Software Encryption - FSE 2014*, Lecture Notes in Computer Science. Springer, 2104.
15. Hoda A. Alkhzaimi and Martin M. Lauridsen. Cryptanalysis of the SIMON Family of Block Ciphers. Cryptology ePrint Archive, Report 2013/543, 2013. `http://eprint.iacr.org/`.
16. Javad Alizadeh, Nasour Bagheri, Praveen Gauravaram, Abhishek Kumar, and Somitra Kumar Sanadhya. Linear Cryptanalysis of Round Reduced SIMON. Cryptology ePrint Archive, Report 2013/663, 2013. `http://eprint.iacr.org/`.
17. Alex Biryukov, Arnab Roy, and Vesselin Velichkov. Differential Analysis of Block Ciphers SIMON and SPECK. In Carlos Cid and Christian Rechberger, editors, *International Workshop on Fast Software Encryption - FSE 2014*, Lecture Notes in Computer Science. Springer, 2104.
18. Alex Biryukov and Vesselin Velichkov. Automatic Search for Differential Trails in ARX Ciphers. In Josh Benaloh, editor, *CT-RSA*, volume 8366 of *Lecture Notes in Computer Science*, pages 227–250. Springer, 2014.
19. Ning Wang, Xiaoyun Wang, Keting Jia, and Jingyuan Zhao. Improved Differential Attacks on Reduced SIMON Versions. Cryptology ePrint Archive, Report 2014/448, 2014. `http://eprint.iacr.org/`.
20. Farzaneh Abed, Eik List, Stefan Lucks, and Jakob Wenzel. Differential and Linear Cryptanalysis of Reduced-Round Simon. Cryptology ePrint Archive, Report 2013/526, 2013. `http://eprint.iacr.org/`.
21. Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The Block Cipher Square. In Eli Biham, editor, *FSE*, volume 1267 of *Lecture Notes in Computer Science*, pages 149–165. Springer, 1997.

22. Lars R. Knudsen and David Wagner. Integral Cryptanalysis. In Joan Daemen and Vincent Rijmen, editors, *FSE*, volume 2365 of *Lecture Notes in Computer Science*, pages 112–127. Springer, 2002.
23. Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David Wagner, and Doug Whiting. Improved Cryptanalysis of Rijndael. In Bruce Schneier, editor, *FSE*, volume 1978 of *Lecture Notes in Computer Science*, pages 213–230. Springer, 2000.
24. Yu Sasaki and Lei Wang. Meet-in-the-Middle Technique for Integral Attacks against Feistel Ciphers. In Knudsen and Wu [29], pages 234–251.
25. Yu Sasaki and Lei Wang. Bitwise Partial-sum on HIGHT: A New Tool for Integral Analysis against ARX Designs. In *ICISC*, Lecture Notes in Computer Science. Springer, 2013.
26. Andrey Bogdanov and Vincent Rijmen. Linear hulls with correlation zero and linear cryptanalysis of block ciphers. *Des. Codes Cryptography*, 70(3):369–383, 2014.
27. 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.
28. Bart Preneel and Tsuyoshi Takagi, editors. *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*. Springer, 2011.
29. Lars R. Knudsen and Huapeng Wu, editors. *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, volume 7707 of *Lecture Notes in Computer Science*. Springer, 2013.

A Round Function of SIMON

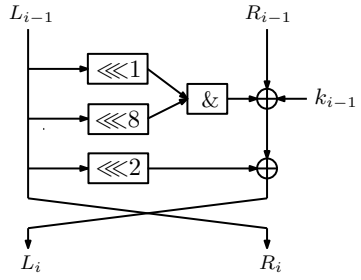


Fig. 6. The Round Function of SIMON

B Details of Zero-Correlation Linear Cryptanalysis

Table 4. Zero-Correlation Linear Approximations of 12-round SIMON48

	Round	Left	Right
Forward	0	0000,0000,0000,0000,0000,0001	0000,0000,0000,0000,0000,0000
	1	0000,0000,0000,0000,0000,0000	0000,0000,0000,0000,0000,0001
	2	0000,0000,0000,0000,0000,0001	?100,000?,0000,0000,0000,0000
	3	?100,000?,0000,0000,0000,0000	0??1,0000,??00,000?,0000,0001
	4	0??1,0000,??00,000?,0000,0001	?0??,?10?,0???,0000,??00,000?
	5	?0??,?10?,0???,0000,??00,000?	????,????,????,??0?,0???,0001
	6	????,????,????,??0?,0???,0001	????,????,????,????,????,??0?
	7	????,????,????,????,????,??0?	????,????,????,????,????,????
Backward	5	????,????,????,?0?0,???0,001?	0???,10?0,???0,000?,?000,00??
	4	0???,10?0,???0,000?,?000,00??	??10,000?,?000,00?0,0000,0010
	3	??10,000?,?000,00?0,0000,0010	1000,00?0,0000,0000,0000,000?
	2	1000,00?0,0000,0000,0000,000?	0000,0000,0000,0000,0000,0010
	1	0000,0000,0000,0000,0000,0010	0000,0000,0000,0000,0000,0000
	0	0000,0000,0000,0000,0000,0000	0000,0000,0000,0000,0000,0010

Table 5. Attack Procedure in Step 1

i	Input state (S_i^1)	Guessed subkey bit	Output state S_{i+1}^1	Counters related to S_{i+1}^1
0	$L_{0,\{0\sim15\}} \| R_{0,\{0\sim15\}}$	$k_{16,\{7\}} \| k_{17,\{5\sim6,15\}} \| k_{18,\{3\sim5,7,13\sim14\}} \| k_{19,\{1\sim6,11\sim13,15\}}$	$L_{0,\{0\sim15\}} \| R_{0,\{0,2\sim14\}} \| R_{16,\{7\}}$	T_{1,S_1^1}
1	$L_{0,\{0\sim15\}} \| R_{0,\{0,2\sim14\}} \| R_{16,\{7\}}$	$k_{0,\{2,8\}}$	$L_{0,\{1\sim15\}} \| R_{0,\{0,3\sim7,9\sim14\}} \| L_{1,\{2,8\}} \| R_{16,\{7\}}$	T_{2,S_2^1}
2	$L_{0,\{1\sim15\}} \| R_{0,\{0,3\sim7,9\sim14\}} \| L_{1,\{2,8\}} \| R_{16,\{7\}}$	$k_{0,\{9\}}$	$L_{0,\{1\sim6,8\sim15\}} \| R_{0,\{0,3\sim7,10\sim14\}} \| L_{1,\{2,8\sim9\}} \| R_{16,\{7\}}$	T_{3,S_3^1}
3	$L_{0,\{1\sim6,8\sim15\}} \| R_{0,\{0,3\sim7,10\sim14\}} \| L_{1,\{2,8\sim9\}} \| R_{16,\{7\}}$	$k_{0,\{3\}}$	$L_{0,\{2\sim6,8\sim15\}} \| R_{0,\{0,4\sim7,10\sim14\}} \| L_{1,\{2\sim3,8\sim9\}} \| R_{16,\{7\}}$	T_{4,S_4^1}
4	$L_{0,\{2\sim6,8\sim15\}} \| R_{0,\{0,4\sim7,10\sim14\}} \| L_{1,\{2\sim3,8\sim9\}} \| R_{16,\{7\}}$	$k_{0,\{4,10\}}$	$L_{0,\{3\sim6,8\sim15\}} \| R_{0,\{0,5\sim7,11\sim14\}} \| L_{1,\{2\sim4,8\sim10\}} \| R_{16,\{7\}}$	T_{5,S_5^1}
5	$L_{0,\{3\sim6,8\sim15\}} \| R_{0,\{0,5\sim7,11\sim14\}} \| L_{1,\{2\sim4,8\sim10\}} \| R_{16,\{7\}}$	$k_{0,\{11\}}$	$L_{0,\{3\sim6,8,10\sim15\}} \| R_{0,\{0,5\sim7,12\sim14\}} \| L_{1,\{2\sim4,8\sim11\}} \| R_{16,\{7\}}$	T_{6,S_6^1}
6	$L_{0,\{3\sim6,8,10\sim15\}} \| R_{0,\{0,5\sim7,12\sim14\}} \| L_{1,\{2\sim4,8\sim11\}} \| R_{16,\{7\}}$	$k_{0,\{5\}}$	$L_{0,\{4\sim6,8,10\sim15\}} \| R_{0,\{0,6\sim7,12\sim14\}} \| L_{1,\{2\sim5,8\sim11\}} \| R_{16,\{7\}}$	T_{7,S_7^1}
7	$L_{0,\{4\sim6,8,10\sim15\}} \| R_{0,\{0,6\sim7,12\sim14\}} \| L_{1,\{2\sim5,8\sim11\}} \| R_{16,\{7\}}$	$k_{0,\{0,6\sim7,12\sim14\}}$	$L_{1,\{0,2\sim14\}} \| R_{1,\{4\sim6,8,10\sim15\}} \| R_{16,\{7\}}$ (also denoted as S_0^2)	T_{8,S_8^1} (T_{8,S_0^2})

The time complexities of substeps $0-7$ are estimated as follows:
substep 0: $2^{20}\cdot 2^{32}\cdot 4/20 = 2^{52}/5$; substep 1: $2^{20}\cdot 2^{31}\cdot 2^2\cdot 2/(16\cdot 20) = 2^{48}/5$;
substep 2: $2^{20}\cdot 2^{30}\cdot 2^3/(16\cdot 20) = 2^{47}/5$; substep 3: $2^{20}\cdot 2^{29}\cdot 2^4/(16\cdot 20) = 2^{47}/5$;
substep 4: $2^{20}\cdot 2^{28}\cdot 2^6\cdot 2/(16\cdot 20) = 2^{49}/5$; substep 5: $2^{20}\cdot 2^{27}\cdot 2^7/(16\cdot 20) = 2^{48}/5$;
substep 6: $2^{20}\cdot 2^{26}\cdot 2^8/(16\cdot 20) = 2^{48}/5$; substep 7: $2^{20}\cdot 2^{25}\cdot 2^{14}\cdot 6/(16\cdot 20) = 2^{54}\cdot 3/5$.

Chapter 12

Links Among Impossible Differential, Integral and Zero Correlation Linear Cryptanalysis

Publication Data

B. Sun, Z. Liu, V. Rijmen, R. Li, L. Cheng, Q. Wang, H. Alkhzaimi, C. Li: Links Among Impossible Differential, Integral and Zero Correlation Linear Cryptanalysis. In R. Gennaro and M.J.B. Robshaw (Eds.): *CRYPTO* 2015, Part I, volumn 9215 of *Lecture Notes in Computer Science*, pages 95–115, 2015.

Contributions

Major contributor of Section 3 and Section 6:

- The idea of theorem is from Bing Sun, I give the proof in Section 3, and write an early version of the text.
- Work on the four new integral distinguishers in Section 6.

Links among Impossible Differential, Integral and Zero Correlation Linear Cryptanalysis ⋆

Bing Sun[1,3], Zhiqiang Liu[2,3,⋆⋆⋆], Vincent Rijmen[3,⋆⋆⋆], Ruilin Li[4,⋆⋆⋆], Lei Cheng[1], Qingju Wang[2,3], Hoda Alkhzaimi[5], Chao Li[1]

[1] College of Science, National University of Defense Technology, Changsha, Hunan, P.R.China, 410073
[2] Dept. Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, P.R.China, 200240
[3] Dept. Electrical Engineering (ESAT), KU Leuven and iMinds, Belgium
[4] College of Electronic Science and Engineering, National University of Defense Technology, Changsha, Hunan, P.R.China, 410073
[5] Technical University of Denmark, Denmark
happy_come@163.com, ilu_zq@sjtu.edu.cn, vincent.rijmen@esat.kuleuven.be, securitylrl@163.com

Abstract. As two important cryptanalytic methods, impossible differential and integral cryptanalysis have attracted much attention in recent years. Although relations among other cryptanalytic approaches have been investigated, the link between these two methods has been missing. The motivation in this paper is to fix this gap and establish links between impossible differential cryptanalysis and integral cryptanalysis.

Firstly, by introducing the concept of structure and dual structure, we prove that $a \to b$ is an impossible differential of a structure $\mathcal{E}$ if and only if it is a zero correlation linear hull of the dual structure $\mathcal{E}^{\perp}$. Meanwhile, our proof shows that the automatic search tool presented by Wu and Wang could find all impossible differentials of both Feistel structures with SP-type round functions and SPN structures. Secondly, by establishing some boolean equations, we show that a zero correlation linear hull always indicates the existence of an integral distinguisher. With this observation we improve the number of rounds of integral distinguishers of Feistel structures, CAST-256, SMS4 and Camellia. Finally, we conclude that an r-round impossible differential of $\mathcal{E}$ always leads to an r-round integral distinguisher of the dual structure $\mathcal{E}^{\perp}$. In the case that $\mathcal{E}$ and $\mathcal{E}^{\perp}$ are linearly equivalent, we derive a direct link between impossible differentials and integral distinguishers of $\mathcal{E}$.

Our results could help to classify different cryptanalytic tools and facilitate the task of evaluating security of block ciphers against various cryptanalytic approaches.

Keywords: Impossible Differential, Integral, Zero Correlation Linear, Feistel, SPN, Camellia, CAST-256, SMS4, PRESENT, PRINCE, ARIA

⋆ The work in this paper is supported by the National Natural Science Foundation of China(No: 61202371, 61402515, 61472250), and National Basic Research Program of China (973 Program) (2013CB338002, 2013CB338004).

⋆⋆⋆ Corresponding Authors

1 Introduction

Block ciphers are considered vital elements in constructing many symmetric cryptographic schemes such as encryption algorithms, hash functions, authentication schemes and pseudo-random number generators. The core security of these schemes depends on the resistance of the underlying block ciphers to known cryptanalytic techniques. So far a variety of cryptanalytic techniques have been proposed such as impossible differential cryptanalysis [1, 2], integral cryptanalysis [3], zero correlation linear cryptanalysis [4], etc.

Impossible differential cryptanalysis was independently proposed by Knudsen [1] and Biham [2]. One of the most popular impossible differentials is called a truncated impossible differential. It is independent of the choices of the S-boxes. Several approaches have been proposed to derive truncated impossible differentials of a block cipher/structure effectively such as the $\mathcal{U}$-method [5], *UID*-method [6] and the extended tool of the former two methods generalized by Wu and Wang in Indocrypt 2012 [7]. Integral cryptanalysis [3] was first proposed by Knudsen and Wagner, and a number of these ideas have been exploited, such as square attack [8], saturation attack [9], multi-set attack [10], and higher order differential attack [11, 12]. With some special inputs, we check whether the sum of the corresponding ciphertexts is zero or not. Usually, we do not need to investigate the details of the S-boxes and only view the S-boxes as some bijective transformations over finite fields. Zero correlation linear cryptanalysis, proposed by Bogdanov and Rijmen in [4], tries to construct some linear hulls with correlation exactly zero. In most cases, as in impossible differential and integral cryptanalysis, we do not need to investigate the details of the S-boxes. Generally, though there has been lots of work concentrating on the design and cryptanalysis of S-boxes [13], most cryptanalytic results by using impossible differential, integral and zero correlation linear cryptanalysis are independent of the choices of the S-boxes. If we choose some other S-boxes in a cipher, the corresponding cryptanalytic results will remain almost the same.

Along with the growing of the list of cryptanalytic tools, the question whether there are direct links or any connections among different tools has drawn much attention of the cryptographic research community, since such relations can be used to compare the effectiveness of different tools as well as to improve cryptanalytic results on block ciphers.

Efforts to find and build the links among different cryptanalytic techniques were initiated by Chabaud and Vaudenay in [14], where a theoretical link between differential and linear cryptanalysis was presented. After that, many attempts have been made to establish further relations among various cryptanalytic tools. In [15], Sun *et al.* proved that from an algebraic view, integral cryptanalysis can be seen as a special case of the interpolation attack. In [16], Leander stated that statistical saturation distinguishers are averagely equivalent to multidimensional linear distinguishers. In [17], Bogdanov *et al.* showed that an integral implies a zero correlation linear hull unconditionally, a zero correlation linear hull indicates an integral distinguisher under certain conditions, and a zero correlation linear hull is actually a special case of multidimensional linear distinguishers. In [18],

Blondeau and Nyberg further analyzed the link between differential and linear cryptanalysis and demonstrated some new insights on this link to make it more applicable in practice. They established new formulas between the probability of truncated differentials and the correlation of linear hulls. This link was later applied in [19] to provide an exact expression of the bias of a differential-linear approximation. Moreover, they claimed that the existence of a zero correlation linear hull is equivalent to the existence of an impossible differential in some specific cases [18]. As shown in [20], this link is usually not practical for most known impossible differential or zero correlation linear distinguishers, since the sum of the dimensions of input and output of each distinguisher is always the block size of the cipher, which means if the dimension parameter for one type is small, it will be infeasibly large for the other type. Blondeau *et al.* proposed a practical relation between these two distinguishers for Feistel-type and Skipjack-type ciphers and showed some equivalence between impossible differentials and zero correlation linear hulls with respect to Feistel-type and Skipjack-type ciphers [20]. In [21], Blondeau and Nyberg gave the link between truncated differential and multidimensional linear approximation, and then applied this link to explore the relations between the complexities of chosen-plaintext and known-plaintext distinguishing/key recovery attacks of differential and linear types. Moreover, they showed that statistical saturation cryptanalysis is indeed equivalent to truncated differential cryptanalysis, which could be used to estimate the data requirement of the statistical saturation key recovery attack.

Contributions. Although there have been intriguing results with respect to the relations among some important cryptanalytic approaches, the link between impossible differential cryptanalysis and integral cryptanalysis is still missing. In this paper, we aim to explore the link between these two cryptanalytic methods. Since the fundamental step in statistical cryptanalysis of block ciphers is to construct effective distinguishers, we focus on building the links among impossible differential, zero correlation linear and integral cryptanalysis from the aspect of distinguishers. Our main contributions are as follows (see Fig.1).

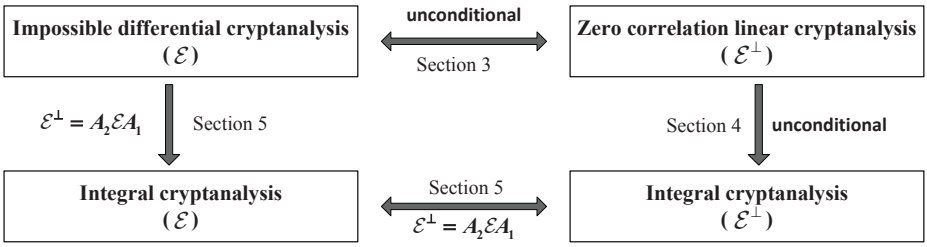


Fig. 1. Links among Impossible Differential, Integral and Zero Correlation Linear Cryptanalysis, where $\mathcal{E}$ is a structure and $\mathcal{E}^{\perp}$ is the dual structure of $\mathcal{E}$, A_1 and A_2 are linear transformations applied before the input and after the output of $\mathcal{E}$.

1. We characterize what "being independent of the choices of S-boxes" means by proposing the definition of *structure* $\mathcal{E}$, which is a set containing some ciphers that are "similar" to each other. Then, by introducing the *dual structure* $\mathcal{E}^{\perp}$, we prove that $a \to b$ is an impossible differential of $\mathcal{E}$ if and only if it is a zero correlation linear hull of $\mathcal{E}^{\perp}$. More specifically, let P^T and P^{-1} denote the transpose and inverse of P respectively. Then for a Feistel structure with SP-type round functions where P is invertible, denoted as $\mathcal{F}_{SP}$, constructing an r-round zero correlation linear hull is equivalent to constructing an impossible differential of $\mathcal{F}_{SP^T}$, which is the same structure as $\mathcal{F}_{SP}$ with P^T instead of P; For an SPN structure $\mathcal{E}_{SP}$, constructing an r-round zero correlation linear hull of $\mathcal{E}_{SP}$ is equivalent to constructing an impossible differential of $\mathcal{E}_{S(P^{-1})^T}$, which is the same structure as $\mathcal{E}_{SP}$ with $(P^{-1})^T$ instead of P. Based on this result, we find 8-round zero correlation linear hulls of Camellia without FL/FL^{-1} layer and 4-round zero correlation linear hulls of ARIA.
2. We show that the automatic search tool, presented by Wu and Wang in Indocrypt 2012, could find all impossible differentials of a cipher that are independent of the choices of the S-boxes. This can be used in provable security of block ciphers against impossible differential cryptanalysis.
3. We find that a zero correlation linear hull always implies the existence of an integral distinguisher, which means the conditions used for deriving integral distinguisher from zero correlation linear hull in [17] can be removed. Meanwhile, we observe that the statement "*integral unconditionally implies zero correlation linear hull*" in [17] is correct only under the definition that integral property is a balanced vectorial boolean function, while it does not hold for the general case. For example, up to date we cannot use the integral distinguisher for 4-round AES (with extra MixColumns) [4, 8] to construct a zero correlation linear hull.
4. Following the results given above, we build the link between impossible differential cryptanalysis and integral cryptanalysis, i.e., an r-round impossible differential of a structure $\mathcal{E}$ always implies the existence of an r-round integral distinguisher of $\mathcal{E}^{\perp}$. Moreover, in the case that $\mathcal{E}^{\perp} = A_2\mathcal{E}A_1$ where A_1 and A_2 are linear transformations, we could get direct links between impossible differential cryptanalysis and integral cryptanalysis of $\mathcal{E}$. Specifically, an r-round impossible differential of SPN structure which adopts bit permutation as the linear layer, always leads to an r-round integral distinguisher.
5. We improve the integrals of Feistel structures by 1 round, build a 24-round integral of CAST-256, present a 12-round integral of SMS4 which is 2-round longer than previously best known ones, and construct an 8-round integral for Camellia without FL/FL^{-1} layers. These distinguishers could not be obtained by the known methods for constructing integral distinguishers or by using the link given in [17]. As an example, the best known key recovery attack on reduced round CAST-256 in non-weak key model is given to show the effectiveness of the newly constructed distinguishers.

Organization. The remainder of this paper is organized as follows. Sec. 2 introduces the notations and concepts that will be used throughout the paper. In Sec. 3, we establish the new links between impossible differential and zero correlation linear cryptanalysis. Sec. 4 shows the refined link between integral and zero correlation linear cryptanalysis. The link between impossible differential and integral cryptanalysis is presented in Sec. 5. Then in Sec. 6, we give some examples to show the effectiveness of the newly established links in constructing new distinguishers of block ciphers. Finally, Sec. 7 concludes this paper.

2 Preliminaries

2.1 Boolean Functions

This section recalls the notations and concepts [22] which will be used throughout this paper. Let $\mathbb{F}_2$ denote the finite field with two elements, and $\mathbb{F}_2^n$ be the vector space over $\mathbb{F}_2$ with dimension n. Let $a = (a_1, \ldots, a_n), b = (b_1, \ldots, b_n) \in \mathbb{F}_2^n$. Then

$$a \cdot b \triangleq a_1 b_1 \oplus \cdots \oplus a_n b_n$$

denotes the *inner product* of a and b. Note that the inner product of a and b can be written as ab^T where b^T stands for the *transpose* of b and the multiplication is defined as matrix multiplication. Given a function $G : \mathbb{F}_2^n \to \mathbb{F}_2$, the *correlation* of G is defined by

$$c(G(x)) \triangleq \frac{\#\{x \in \mathbb{F}_2^n | G(x) = 0\} - \#\{x \in \mathbb{F}_2^n | G(x) = 1\}}{2^n} = \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} (-1)^{G(x)}.$$

Given a vectorial function $H : \mathbb{F}_2^n \to \mathbb{F}_2^k$, the *correlation* of the linear approximation for a k-bit output mask b and an n-bit input mask a is defined by

$$c(a \cdot x \oplus b \cdot H(x)) \triangleq \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} (-1)^{a \cdot x \oplus b \cdot H(x)}.$$

If $c(a \cdot x \oplus b \cdot H(x)) = 0$, then $a \to b$ is called a *zero correlation linear hull* of H[4]. This definition can be extended as follows: Let $A \subseteq \mathbb{F}_2^n$, $B \subseteq \mathbb{F}_2^k$. If for all $a \in A$, $b \in B$, $c(a \cdot x \oplus b \cdot H(x)) = 0$, then $A \to B$ is called a *zero correlation linear hull* of H. In the case that H is a permutation on $\mathbb{F}_2^n$, for any $b \neq 0$, $c(b \cdot H(x)) = 0$ and for any $a \neq 0$, $c(a \cdot x) = 0$. We call $0 \to b$ and $a \to 0$ trivial zero correlation linear hulls of H where $a \neq 0$ and $b \neq 0$. Let $A \subseteq \mathbb{F}_2^n$. If the size of the set

$$H_A^{-1}(y) \triangleq \{x \in A | H(x) = y\}$$

is independent of $y \in \mathbb{F}_2^k$, we say H is *balanced on* A. Specifically, if $A = \mathbb{F}_2^n$, we say H is a *balanced function*. If the sum of all images of H is 0, i.e.

$$\sum_{x \in \mathbb{F}_2^n} H(x) = 0,$$

we say H has an *integral-balanced (zero-sum)* property[3]. Let $\delta \in \mathbb{F}_2^n$ and $\Delta \in \mathbb{F}_2^k$. The differential probability of $\delta \to \Delta$ is defined as

$$p(\delta \to \Delta) \triangleq \frac{\#\{x \in \mathbb{F}_2^n | H(x) \oplus H(x \oplus \delta) = \Delta\}}{2^n}.$$

If $p(\delta \to \Delta) = 0$, then $\delta \to \Delta$ is called an *impossible differential* of H[1, 2]. Let $A \subseteq \mathbb{F}_2^n$, $B \subseteq \mathbb{F}_2^k$. If for all $a \in A$ and $b \in B$, $p(a \to b) = 0$, $A \to B$ is called an *impossible differential* of H. We recall the following property of balanced boolean functions: a function $G : \mathbb{F}_2^n \to \mathbb{F}_2$ is balanced if and only if $c(G(x)) = 0$.

2.2 Block Ciphers

Feistel Ciphers. An r-round Feistel cipher E is defined as follows: Let $(L_0, R_0) \in \mathbb{F}_2^{2n}$ be the input of E. Iterate the following transformation r times:

$$\begin{cases} L_{i+1} = F_i(L_i) \oplus R_i \\ R_{i+1} = L_i \end{cases} \quad 0 \le i \le r-1,$$

where $L_i, R_i \in \mathbb{F}_2^n$. The output of the r-th iteration is defined as the output of E. In this paper, we will focus on the case that F_i's are SP-type functions which will be defined in the following.

SPN Ciphers. The SPN structure is widely used in constructing cryptographic primitives. It iterates some SP-type round functions to achieve confusion and diffusion. Specifically, the SP-type function $f : \mathbb{F}_2^{s \times t} \to \mathbb{F}_2^{s \times t}$ used in this paper is defined as follows: Assume the input x is divided into t pieces $x = (x_0, \ldots, x_{t-1})$, and each of the x_i's is an s-bit word. Then apply the nonlinear transformation S_i to x_i and let $y = (S_0(x_0), \ldots, S_{t-1}(x_{t-1})) \in \mathbb{F}_2^{s \times t}$. At last, apply a linear transformation P to y, and Py is the output of f.

The following strategies are popular in designing the diffusion layer P of a cipher:

(1) P is a bit-wise permutation of $\mathbb{F}_2^{s \times t}$ as in PRESENT [23]. PRESENT adopts bit permutation as the diffusion layer P, which can be defined as a permutation matrix $P = (P_{i,j})_{64 \times 64}$:

$$P_{i,j} = \begin{cases} 1 & \text{if } j = 16i \bmod 63 \\ 0 & \text{otherwise} \end{cases}.$$

(2) Each bit of Py is a sum of some bits of y as in PRINCE [24]. Firstly, we will define SR and M' as follows:

SR permutes the 16 nibbles, therefore it is a permutation of 64 bits and we could write SR as a permutation matrix in $\mathbb{F}_2^{64 \times 64}$.

To construct M', we first define

$$\hat{M}^{(0)} = \begin{pmatrix} M_0 & M_1 & M_2 & M_3 \\ M_1 & M_2 & M_3 & M_0 \\ M_2 & M_3 & M_0 & M_1 \\ M_3 & M_0 & M_1 & M_2 \end{pmatrix}, \quad \hat{M}^{(1)} = \begin{pmatrix} M_1 & M_2 & M_3 & M_0 \\ M_2 & M_3 & M_0 & M_1 \\ M_3 & M_0 & M_1 & M_2 \\ M_0 & M_1 & M_2 & M_3 \end{pmatrix}$$

where

$$M_0 = \begin{pmatrix} 0\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{pmatrix}, M_1 = \begin{pmatrix} 1\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{pmatrix}, M_2 = \begin{pmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,0\,1 \end{pmatrix}, M_3 = \begin{pmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,0 \end{pmatrix},$$

and then we define $M' = \mathrm{diag}(\hat{M}^{(0)}, \hat{M}^{(1)}, \hat{M}^{(1)}, \hat{M}^{(0)})$, which is a 64×64 block diagonal matrix.

M' is used as the linear transformation of the middle round. The transformations $M = SR \circ M'$ and M^{-1} are used before and after the middle round, respectively.

(3) Each word of Py is a sum of some words of y as in Camellia [25] and ARIA [26]. The block cipher Camellia was recommended in the NESSIE block cipher portfolio in 2003 and selected as a new international standard by ISO/IEC in 2005. ARIA is a 128-bit block cipher established as a Korean Standard by the Ministry of Commerce, Industry and Energy in 2004. The linear transformations P_{C} and P_{A} of Camellia and ARIA could be written as follows:

$$P_{\mathrm{C}} = \begin{pmatrix} E\,0\,E\,E\,0\,E\,E\,E \\ E\,E\,0\,E\,E\,0\,E\,E \\ E\,E\,E\,0\,E\,E\,0\,E \\ 0\,E\,E\,E\,E\,E\,E\,0 \\ E\,E\,0\,0\,0\,E\,E\,E \\ 0\,E\,E\,0\,E\,0\,E\,E \\ 0\,0\,E\,E\,E\,E\,0\,E \\ E\,0\,0\,E\,E\,E\,E\,0 \end{pmatrix} \quad P_{\mathrm{A}} = \begin{pmatrix} 0\,0\,0\,E\,E\,0\,E\,0\,E\,E\,0\,0\,0\,E\,E\,0 \\ 0\,0\,E\,0\,0\,E\,0\,E\,E\,E\,0\,0\,E\,0\,0\,E \\ 0\,E\,0\,0\,E\,0\,E\,0\,0\,0\,E\,E\,E\,0\,0\,E \\ E\,0\,0\,0\,0\,E\,0\,E\,0\,0\,E\,E\,0\,E\,E\,0 \\ E\,0\,E\,0\,0\,E\,0\,0\,E\,0\,0\,E\,0\,0\,E\,E \\ 0\,E\,0\,E\,E\,0\,0\,0\,0\,E\,E\,0\,0\,0\,E\,E \\ E\,0\,E\,0\,0\,0\,0\,E\,0\,E\,E\,0\,E\,E\,0\,0 \\ 0\,E\,0\,E\,0\,0\,E\,0\,E\,0\,0\,E\,E\,E\,0\,0 \\ E\,E\,0\,0\,E\,0\,0\,E\,0\,0\,E\,0\,0\,E\,0\,E \\ E\,E\,0\,0\,0\,E\,E\,0\,0\,0\,0\,E\,E\,0\,E\,0 \\ 0\,0\,E\,E\,0\,E\,E\,0\,E\,0\,0\,0\,0\,E\,0\,E \\ 0\,0\,E\,E\,E\,0\,0\,E\,0\,E\,0\,0\,E\,0\,E\,0 \\ 0\,E\,E\,0\,0\,0\,E\,E\,0\,E\,0\,E\,E\,0\,0\,0 \\ E\,0\,0\,E\,0\,0\,E\,E\,E\,0\,E\,0\,0\,E\,0\,0 \\ E\,0\,0\,E\,E\,E\,0\,0\,0\,E\,0\,E\,0\,0\,E\,0 \\ 0\,E\,E\,0\,E\,E\,0\,0\,E\,0\,E\,0\,0\,0\,0\,E \end{pmatrix}$$

where E and 0 denote 8×8 identity and zero matrices, respectively.

(4) Each word of Py, seen as an element of some extension fields of $\mathbb{F}_2$, is a linear combination of some other words of y as in the AES. In the following, we will use the matrix expression of finite fields to show how to write the linear layer of AES as a 128×128 binary matrix:

Since ShiftRows is a permutation on 16 bytes, it is also a permutation on 128 bits. Therefore, as in the discussion above, we can represent ShiftRows as a permutation matrix M_{SR} in $\mathbb{F}_2^{128\times128}$. Let $\mathbb{F}_{2^8} = \mathbb{F}_2[x]/ < f(x) >$ where $\mathbb{F}_2[x]$ is the polynomial ring over $\mathbb{F}_2$, $f(x) = x^8 + x^4 + x^3 + x + 1 \in \mathbb{F}_2[x]$ is the defining polynomial of $\mathbb{F}_{2^8}$. Then $1 = (00000001) \in \mathbb{F}_{2^8}$ can be written as the 8×8 identity matrix E, $2 = (00000010) \in \mathbb{F}_{2^8}$ can be written as the following 8×8

matrix:

$$M_2 = \begin{pmatrix} 0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&1 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&1 \\ 0&0&0&1&0&0&0&1 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \end{pmatrix}$$

and the matrix representation of $3 = (00000011)$ is $M_3 = E \oplus M_2$. If we substitute 1, 2 and 3 in MixColumns by E, M_2 and M_3, respectively, we get a 128×128 binary matrix M_{MC} and the linear layer of AES can be written as $M_{MC}M_{SR}$ which is a 128×128 matrix over $\mathbb{F}_2$.

Generally, no matter which linear transformation a cipher adopts, it is always linear over $\mathbb{F}_2$. Therefore, P can always be written as a multiplication by a matrix which leads to the following definition:

Definition 1. *Let P be a linear transformation over $\mathbb{F}_2^m$ for some positive integer m. The matrix representation of P over $\mathbb{F}_2$ is called the* primitive representation *of P.*

2.3 Structure and Dual Structure

In many cases, when constructing impossible differentials and zero correlation linear hulls, we are only interested in detecting whether there is a difference (mask) of an S-box or not, regardless of the value of this difference (mask). For example, the truncated impossible differential and zero correlation linear hull of AES in [4, 27] and Camellia in [28, 29]. In other words, if these ciphers adopt some other S-boxes, these distinguishers still hold. This leads to the following definition:

Definition 2. *Let $E : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a block cipher with bijective S-boxes as the basic non-linear components.*

(1) *A* structure *$\mathcal{E}^E$ on $\mathbb{F}_2^n$ is defined as a set of block ciphers E' which is exactly the same as E except that the S-boxes can take all possible bijective transformations on the corresponding domains.*
(2) *Let $a, b \in \mathbb{F}_2^n$. If for any $E' \in \mathcal{E}^E$, $a \to b$ is an impossible differential (zero correlation linear hull) of E', $a \to b$ is called an impossible differential (zero correlation linear hull) of $\mathcal{E}^E$.*

Note. In the definition of $\mathcal{E}^E$, if E uses bijective S-boxes, then the S-boxes in $\mathcal{E}^E$ should be bijective. However, if S-boxes used in E are not necessarily bijective, then $\mathcal{E}^E$ could be defined as a set of block ciphers E' which is exactly the same as E except that the S-boxes can take all possible transformations on the corresponding domains. As discussed above, the truncated impossible

differentials and zero correlation linear hulls of AES and Camellia found so far are actually the impossible differentials and zero correlation linear hulls of $\mathcal{E}^{\mathrm{AES}}$ and $\mathcal{E}^{\mathrm{Camellia}}$.

Definition 3. *Let $\mathcal{F}_{SP}$ be a* Feistel structure *with SP-type round function, and let the primitive representation of the linear transformation be P. Let σ be the operation that exchanges the left and right halves of a state. Then the* dual structure *$\mathcal{F}_{SP}^{\perp}$ of $\mathcal{F}_{SP}$ is defined as $\sigma \circ \mathcal{F}_{P^T S} \circ \sigma$.*

Let $\mathcal{E}_{SP}$ be an SPN structure *with primitive representation of the linear transformation being P. Then the* dual structure *$\mathcal{E}_{SP}^{\perp}$ of $\mathcal{E}_{SP}$ is defined as $\mathcal{E}_{S(P^{-1})^T}$.*

3 Links between Impossible Differential and Zero Correlation Linear Cryptanalysis

In this section, we will show the equivalence between impossible differentials and zero correlation linear hulls of a structure, which will be used to establish the link between impossible differential and integral cryptanalysis in Sec. 5. The next theorem is stated without proof in [17].

Theorem 1. *$a \to b$ is an r-round impossible differential of $\mathcal{F}_{SP}$ if and only if it is an r-round zero correlation linear hull of $\mathcal{F}_{SP}^{\perp}$.*

Proof. The proof can be divided into the following two parts (See Fig.2):

Part (I) We prove that for $(\delta_0, \delta_1) \to (\delta_r, \delta_{r+1})$, if one can find $E \in \mathcal{F}_{SP}^{\perp}$ such that $c((\delta_0, \delta_1) \cdot x \oplus (\delta_r, \delta_{r+1}) \cdot E(x)) \neq 0$, then one can find $E' \in \mathcal{F}_{SP}$ such that $p((\delta_1, \delta_0) \to (\delta_{r+1}, \delta_r)) > 0$.

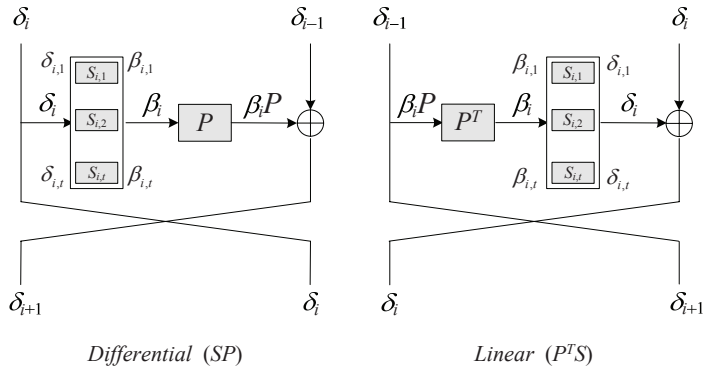


Fig. 2. Differential Propagation of $\mathcal{F}_{SP}$ and Linear Propagation of $\mathcal{F}_{SP}^{\perp}$

Assume that $(\delta_0, \delta_1) \to (\delta_r, \delta_{r+1})$ is a linear hull with non-zero correlation for some $E \in \mathcal{F}_{SP}^{\perp}$, and the input to the round function could be divided into t

pieces, each of which is an s-bit word. Then there exists a linear characteristic with non-zero correlation:

$$(\delta_0, \delta_1) \to \cdots (\delta_{i-1}, \delta_i) \to \cdots \to (\delta_r, \delta_{r+1}),$$

where $\delta_i \in (\mathbb{F}_2^s)^t$. In this characteristic, the output mask of $S_i = (S_{i,1}, \ldots, S_{i,t})$ is $\delta_i = (\delta_{i,1}, \ldots, \delta_{i,t}) \in (\mathbb{F}_2^s)^t$, and let the input mask of S_i be $\beta_i = (\beta_{i,1}, \ldots, \beta_{i,t}) \in (\mathbb{F}_2^s)^t$. Since for $\gamma \neq \beta_i P$, $c(\gamma \cdot x \oplus \beta_i \cdot (xP^T)) = 0$, $\delta_{i+1} = \delta_{i-1} \oplus \beta_i P$.

In the following, for any $(x_L, x_R) = (x_{L,1}, \ldots, x_{L,t}, x_{R,1}, \ldots, x_{R,t}) \in (\mathbb{F}_2^s)^t \times (\mathbb{F}_2^s)^t$, we will construct an r-round cipher $E_r \in \mathcal{F}_{SP}$, such that $E_r(x_L, x_R) \oplus E_r(x_L \oplus \delta_1, x_R \oplus \delta_0) = (\delta_{r+1}, \delta_r)$.

If $r = 1$, for $j \in \{1, \ldots, t\}$: if $\delta_{1,j} = 0$, we can define $S_{1,j}$ as any possible transformation on $\mathbb{F}_2^s$, and if $\delta_{1,j} \neq 0$, we can define

$$S_{1,j}(x_{L,j}) = x_{L,j}, \quad S_{1,j}(x_{L,j} \oplus \delta_{1,j}) = x_{L,j} \oplus \beta_{1,j},$$

then for $E_1 \in \mathcal{F}_{SP}$ which adopts such S-boxes,

$$E_1(x_L, x_R) \oplus E_1(x_L \oplus \delta_1, x_R \oplus \delta_0) = (\delta_0 \oplus \beta_1 P, \delta_1) = (\delta_2, \delta_1).$$

Suppose that we have constructed E_{r-1} such that $E_{r-1}(x_L, x_R) \oplus E_{r-1}(x_L \oplus \delta_1, x_R \oplus \delta_0) = (\delta_r, \delta_{r-1})$. Denote by $(y_L, y_R) = (y_{L,1}, \ldots, y_{L,t}, y_{R,1}, \ldots, y_{R,t})$ the output of $E_{r-1}(x_L, x_R)$. Then in the r-th round, if $\delta_{r,j} = 0$, we can define $S_{r,j}$ as any possible transformation on $\mathbb{F}_2^s$, otherwise, define $S_{r,j}$ as follows:

$$S_{r,j}(y_{L,j}) = y_{L,j}, \quad S_{r,j}(y_{L,j} \oplus \delta_{r,j}) = y_{L,j} \oplus \beta_{r,j}.$$

Therefore $E_r(x_L, x_R) \oplus E_r(x_L \oplus \delta_1, x_R \oplus \delta_0) = (\delta_{r-1} \oplus \beta_r P, \delta_r) = (\delta_{r+1}, \delta_r)$.

Part (II) We prove that for $(\delta_1, \delta_0) \to (\delta_{r+1}, \delta_r)$, if one can find some $E \in \mathcal{F}_{SP}$ such that $p((\delta_1, \delta_0) \to (\delta_{r+1}, \delta_r)) > 0$, one can find some $E' \in \mathcal{F}_{SP}^{\perp}$ such that $c((\delta_0, \delta_1) \cdot x \oplus (\delta_r, \delta_{r+1}) \cdot E'(x)) \neq 0$.

Assume that $(\delta_1, \delta_0) \to (\delta_{r+1}, \delta_r)$ is a differential of $E \in \mathcal{F}_{SP}$. Then there exists a differential characteristic with positive probability:

$$(\delta_1, \delta_0) \to \cdots (\delta_{i+1}, \delta_i) \to \cdots \to (\delta_{r+1}, \delta_r),$$

where $\delta_i \in (\mathbb{F}_2^s)^t$. In this characteristic, the input difference of $S_i = (S_{i,1}, \ldots, S_{i,t})$ is $\delta_i = (\delta_{i,1}, \ldots, \delta_{i,t}) \in (\mathbb{F}_2^s)^t$, and let the output difference of S_i be $\beta_i = (\beta_{i,1}, \ldots, \beta_{i,t}) \in (\mathbb{F}_2^s)^t$, then $\delta_{i+1} = \delta_{i-1} \oplus (\beta_i P)$.

Taking the following fact into consideration: for $(\delta_{i,j}, \beta_{i,j})$, where $\delta_{i,j} \neq 0$, there always exists an $s \times s$ binary matrix $M_{i,j}$ such that $\beta_{i,j} = \delta_{i,j} M_{i,j}^T$, then for $S_{i,j}(x) = xM_{i,j}$, $c(\beta_{i,j} \cdot x \oplus \delta_{i,j} \cdot S_{i,j}(x)) = 1$.

Now we construct an r-round cipher $E_r \in \mathcal{F}_{SP}^{\perp}$ such that $c((\delta_0, \delta_1) \cdot x \oplus (\delta_r, \delta_{r+1}) \cdot E_r(x)) \neq 0$. If $r = 1$, let $S_{1,j}(x) = xM_{1,j}$ for $\delta_{1,j} \neq 0$ and any linear transformation on $\mathbb{F}_2^s$ otherwise. Then all operations in $E_1 \in \mathcal{F}_{SP}^{\perp}$ are linear over $\mathbb{F}_2$, which implies that there exists a $2st \times 2st$ binary matrix M_1 such that $E_1(x) = xM_1$, and

$$c((\delta_0, \delta_1) \cdot x \oplus (\delta_1, \delta_2) \cdot E_1(x)) = 1.$$

Assume that we have constructed $E_{r-1}(x) = xM_{r-1}$ with M_{r-1} being a $2st \times 2st$ binary matrix such that

$$c((\delta_0, \delta_1) \cdot x \oplus (\delta_{r-1}, \delta_r) \cdot E_{r-1}(x)) = 1,$$

and we can define $S_{r,j}(x)$ in the r-th round similarly, then $E_r(x) = xM_r$ for some $2st \times 2st$ binary matrix M_r, and

$$c((\delta_0, \delta_1) \cdot x \oplus (\delta_r, \delta_{r+1}) \cdot E_r(x)) = 1,$$

which ends our proof. □

Note. In the proof of Theorem 1, the S-boxes we constructed are not necessarily bijective. If we add the bijective condition, Theorem 1 still holds. Since for a bijective S-box, if the correlation is non-zero, $\delta_{1,j} \neq 0$ implies $\beta_{1,j} \neq 0$. Therefore, in Part(I) of the proof, we can further define $S_{1,j}$ as

$$S_{1,j}(x) = \begin{cases} x_{L,j} \oplus \delta_{1,j} & x = x_{L,j} \oplus \beta_{1,j}, \\ x_{L,j} \oplus \beta_{1,j} & x = x_{L,j} \oplus \delta_{1,j}, \\ x & \text{others}, \end{cases}$$

and a similar definition can also be given to $S_{r,j}$. In this case, the S-boxes are invertible. Moreover, for a bijective S-box, if the differential probability is positive, $\delta_{i,j} \neq 0$ implies $\beta_{i,j} \neq 0$, thus in Part (II) of the proof, we can always find a non-singular binary matrix $M_{i,j}$ such that $\beta_{i,j} = \delta_{i,j} M_{i,j}^T$.

Similarly, we can prove the following theorem:

Theorem 2. *$a \to b$ is an r-round impossible differential of $\mathcal{E}_{SP}$ if and only if it is an r-round zero correlation linear hull of $\mathcal{E}_{SP}^{\perp}$.*

Definition 2 implies that the "impossibility" of an impossible differential of a structure can be caused only by a differential $\delta_1 \to \delta_2$ where either $\delta_1 = 0$ or $\delta_2 = 0$ (but not both) over an invertible S-box, or by a differential $0 \to \delta_2$ over a non-invertible S-box. Otherwise, according to the proof of Theorem 1, we can always find an S-box such that $\delta_1 \to \delta_2$ is a possible differential. Therefore, we have the following corollary:

Corollary 1. *The method presented in [7] finds all impossible differentials of $\mathcal{F}_{SP}$ and $\mathcal{E}_{SP}$.*

As a matter of fact, this corollary can be used in the provable security of block ciphers against impossible differential cryptanalysis, since with the help of this corollary, the longest impossible differentials of a given structure could be given.

In case P is invertible, according to the definition of equivalent structures given in [30], we have

$$\mathcal{F}_{P^T S} = \left((P^T)^{-1}, (P^T)^{-1}\right) \mathcal{F}_{SP^T} \left(P^T, P^T\right), \tag{1}$$

which indicates:

Corollary 2. *Let $\mathcal{F}_{SP}$ be a Feistel structure with SP-type round function, and let the primitive representation of the linear transformation be P. If P is invertible, finding zero correlation linear hulls of $\mathcal{F}_{SP}$ is equivalent to finding impossible differentials of $\mathcal{F}_{SP^T}$.*

Example 1. **(8-Round Zero Correlation Linear Hull of Camellia Without FL/FL^{-1})** Let Camellia* denote the cipher which is exactly the same as Camellia without FL/FL^{-1} layer except that P^T is used instead of P. Then we find that, for example:

$$((0,0,0,0,0,0,0,0),(0,0,0,0,a,0,0,0)) \rightarrow ((0,0,0,0,0,0,0,h),(0,0,0,0,0,0,0,0))$$

is an 8-round impossible differential of Camellia*, where a and h denote any non-zero values. Therefore, we can derive an 8-round zero correlation linear distinguisher of Camellia without FL/FL^{-1} layer as shown below:

$$((a,a,0,0,a,0,a,a),(0,0,0,0,0,0,0,0)) \rightarrow ((0,0,0,0,0,0,0,0),(h,0,0,h,0,h,h,h)).$$

Furthermore, if $\mathcal{F}_{SP} = \mathcal{F}_{SP^T}$ and $\mathcal{E}_{SP} = \mathcal{E}_{S(P^{-1})^T}$, we have:

Corollary 3. *For a Feistel structure $\mathcal{F}_{SP}$ with SP-type round function, if P is invertible and $P = P^T$, there is a one-to-one correspondence between impossible differentials and zero correlation linear hulls.*

For an SPN structure $\mathcal{E}_{SP}$, if $P^T P = E$, $a \rightarrow b$ is an impossible differential if and only if it is a zero correlation linear hull.

Example 2. **(4-Round Zero Correlation Linear Hull of ARIA)** Since the linear layer P of ARIA satisfies $P^T P = E$, any impossible differential of $\mathcal{E}^{\mathrm{ARIA}}$ is automatically a zero correlation linear hull of $\mathcal{E}^{\mathrm{ARIA}}$. Therefore, the impossible differentials of 4-round ARIA shown in [28] are also zero correlation linear hulls of 4-round ARIA.

4 Links between Integral and Zero Correlation Linear Cryptanalysis

Firstly, we give two fundamental statements that give links between integral cryptanalysis and zero correlation linear cryptanalysis:

Lemma 1. *Let A be a subspace of $\mathbb{F}_2^n$, $A^\perp = \{x \in \mathbb{F}_2^n | a \cdot x = 0, a \in A\}$ be the dual space of A and $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ be a function on $\mathbb{F}_2^n$. For any $\lambda \in \mathbb{F}_2^n$, $T_\lambda : A^\perp \rightarrow \mathbb{F}_2^n$ is defined as $T_\lambda(x) = F(x \oplus \lambda)$, then for any $b \in \mathbb{F}_2^n$,*

$$\sum_{a \in A} (-1)^{a \cdot \lambda} c(a \cdot x \oplus b \cdot F(x)) = c(b \cdot T_\lambda(x)).$$

Proof.

$$\sum_{a\in A}(-1)^{a\cdot\lambda}c(a\cdot x\oplus b\cdot F(x)) = \sum_{a\in A}(-1)^{a\cdot\lambda}\frac{1}{2^n}\sum_{x\in\mathbb{F}_2^n}(-1)^{a\cdot x\oplus b\cdot F(x)}$$
$$= \frac{1}{2^n}\sum_{x\in\mathbb{F}_2^n}(-1)^{b\cdot F(x)}\sum_{a\in A}(-1)^{a\cdot(\lambda\oplus x)} = \frac{1}{2^n}\sum_{x\in\mathbb{F}_2^n}(-1)^{b\cdot F(x)}|A|\delta_{A^\perp}(\lambda\oplus x)$$
$$= \frac{1}{|A^\perp|}\sum_{y\in A^\perp}(-1)^{b\cdot T_\lambda(y)} = c(b\cdot T_\lambda(x)),$$

where $\delta_{A^\perp}(x) = \begin{cases}1 & x\in A^\perp\\ 0 & x\notin A^\perp\end{cases}$. □

Lemma 2. *Let A be a subspace of $\mathbb{F}_2^n$, $F:\mathbb{F}_2^n\to\mathbb{F}_2^n$, and let $T_\lambda : A^\perp\to\mathbb{F}_2^n$ be defined as $T_\lambda(x) = F(x\oplus\lambda)$ where $\lambda\in\mathbb{F}_2^n$. Then for any $b\in\mathbb{F}_2^n$,*

$$\frac{1}{2^n}\sum_{\lambda\in\mathbb{F}_2^n}(-1)^{b\cdot F(\lambda)}c(b\cdot T_\lambda(x)) = \sum_{a\in A}c^2(a\cdot x\oplus b\cdot F(x)).$$

The proof of Lemma 2 is given in the full version of this paper [31]. The conclusion of [17] that integral unconditionally implies zero correlation linear hull, is correct only under their definition of integral, which requires that $c(b\cdot T_\lambda(x)) = 0$. Under the original, more general definition for an integral distinguisher [3], this conclusion may not hold.

From Lemma 1, we can deduce the following:

Corollary 4. *Let $F:\mathbb{F}_2^n\to\mathbb{F}_2^n$ be a function on $\mathbb{F}_2^n$, and let A be a subspace of $\mathbb{F}_2^n$ and $b\in\mathbb{F}_2^n\setminus\{0\}$. Suppose that $A\to b$ is a zero correlation linear hull of F, then for any $\lambda\in\mathbb{F}_2^n$, $b\cdot F(x\oplus\lambda)$ is balanced on $A^\perp$.*

This corollary states that if the input masks of a zero correlation linear hull form a subspace, then a zero correlation linear hull implies an integral distinguisher. Furthermore, the condition that input masks form a subspace can be removed, which leads to the following result:

Theorem 3. *A nontrivial zero correlation linear hull of a block cipher always implies the existence of an integral distinguisher.*

Proof. Assume that $A\to B$ is a non-trivial zero correlation linear hull of a block cipher E. Then we can choose $0\neq a\in A, 0\neq b\in B$, such that $\{0,a\}\to b$ is also a zero correlation linear hull of E.

Since $V=\{0,a\}$ forms a subspace on $\mathbb{F}_2$, according to Corollary 4, $b\cdot E(x)$ is balanced on $V^\perp$. This implies an integral distinguisher of E. □

Moreover, in the proof of Theorem 3, we can always assume that $0\in A$. Then

1. If A forms a subspace, an integral distinguisher can be constructed from $A \to b$;
2. If A does not form a subspace, we can choose some $A_1 \subset A$ such that A_1 forms a subspace, then an integral distinguisher can be constructed from $A_1 \to b$.

It was stated in [17] that a zero correlation linear hull indicates the existence of an integral distinguisher under certain conditions, while Theorem 3 shows that these conditions can be removed. This results in a more applicable link between zero correlation linear cryptanalysis and integral cryptanalysis.

It can be seen that Theorem 3 also gives us a new approach to find integral distinguishers of block ciphers. More specifically, an r-round zero correlation linear hull can be used to construct an r-round integral distinguisher.

5 Links between Impossible Differential and Integral Cryptanalysis

According to the links given in the previous sections, we establish a link between impossible differential cryptanalysis and integral cryptanalysis:

Theorem 4. *Let $\mathcal{E} \in \{\mathcal{F}_{SP}, \mathcal{E}_{SP}\}$. Then an impossible differential of $\mathcal{E}$ always implies the existence of an integral of $\mathcal{E}^{\perp}$.*

Proof. This can be deduced from the following facts:

- A zero correlation linear hull of $\mathcal{E}^{\perp}$ always implies the existence of an integral of $\mathcal{E}^{\perp}$;
- A zero correlation linear hull of $\mathcal{E}^{\perp}$ could be constructed by constructing an impossible differential of $\mathcal{E}$. □

In case $\mathcal{E}^{\perp} = A_2 \mathcal{E} A_1$ where A_1 and A_2 are linear transformations, we get the direct links between impossible differential and integral cryptanalysis:

Corollary 5. *Let $\mathcal{F}_{SP}$ be a Feistel structure with SP-type round function, and let the primitive representation of the linear transformation be P. If P is invertible and there exists a permutation π on t elements such that for any $(x_0, \ldots, x_{t-1}) \in \mathbb{F}_2^{s \times t}$,*

$$P(x_0, \ldots, x_{t-1}) = \pi^{-1} P^T \pi(x_0, \ldots, x_{t-1}),$$

then for $\mathcal{F}_{SP}$, an impossible differential always implies the existence of an integral distinguisher.

Example 3. SNAKE(2) is a Feistel cipher proposed by Lee and Cha at JWISC'97, please refer to [32, 33] for details. According to [30], the round function of SNAKE(2) can be seen as an SP-type one with the primitive presentation of the matrix being defined as

$$P = \begin{pmatrix} E & E & E & E \\ E & 0 & E & E \\ E & 0 & 0 & E \\ E & 0 & 0 & 0 \end{pmatrix},$$

where E and 0 are the identity and zero matrices of $\mathbb{F}_2^{8\times 8}$, respectively. Let

$$\pi = \begin{pmatrix} 1\,0\,0\,0 \\ 0\,0\,0\,1 \\ 0\,0\,1\,0 \\ 0\,1\,0\,0 \end{pmatrix}.$$

Then we have $P = \pi^{-1}P^T\pi$, therefore, an impossible differential of SNAKE(2), which is independent of the details of the S-boxes, always implies the existence of an integral distinguisher of SNAKE(2).

Corollary 6. *Let $\mathcal{E}_{SP}$ be an SPN structure with the primitive representation of the linear transformation being P. If $P^TP = diag(Q_1, \ldots, Q_t)$, where $Q_i \in \mathbb{F}_2^{s\times s}$, then for $\mathcal{E}_{SP}$, an impossible differential always implies the existence of an integral distinguisher.*

Proof. Firstly, according to Theorem 4, if $P^TP = E$, an impossible differential of $\mathcal{E}_{SP}$ always implies the existence of an integral.

Secondly, for the S-layer of $\mathcal{E}_{SP}$, if we substitute S by applying Q_i to the i-th S-box, according to definition 2, the structure stays identical. Since

$$P \circ (\mathrm{diag}(Q_1, \ldots, Q_t) \circ S) = (P \circ \mathrm{diag}(Q_1, \ldots, Q_t)) \circ S,$$

an SPN structure $\mathcal{E}_{SP}$ is equivalent to an SPN structure $\mathcal{E}_{S(P\circ\mathrm{diag}(Q_1,\ldots,Q_t))}$.

Based on the above two points, we can get the conclusion. □

To show applications of these links, we recall that, an $n\times n$ matrix P is called *orthogonal* if and only if $P^TP = E$, where E is the $n \times n$ identity matrix.

Example 4. We can check that, SR and M' used in PRINCE are orthogonal matrices, therefore

$$M^TM = (SR \circ M')^T(SR \circ M') = E,$$

where E is the 64×64 identity matrix. So all the linear layers used in different rounds of PRINCE are orthogonal based on which we could conclude that any r-round impossible differential of PRINCE which is independent of the choices of the S-boxes implies the existence of an r-round integral distinguisher.

Example 5. Since the linear layer P of ARIA is both symmetric and involutional, e.g. $P = P^{-1} = P^T$, any impossible differential of ARIA which is independent of the choices of S-boxes implies the existence of an integral distinguisher.

Example 6. We can check that P used in PRESENT satisfies $P = (P^{-1})^T$, therefore, an impossible differential, which is independent of the details of the S-boxes, always leads to the existence of an integral distinguisher. In fact, since a permutation matrix P is always orthogonal, we have the following Corollary:

Corollary 7. *For an SPN structure which adopts bit permutation as the diffusion layer, the existence of an r-round impossible differential implies the existence of an r-round integral distinguisher.*

6 New Integrals for Block Ciphers/Structures

6.1 New Integrals for Feistel Structures

Let $\mathcal{E}_r$ be an r-round Feistel structure $\mathcal{F}_{SP}$. Then for any $a \neq 0$, $b \neq a$, $(a, 0) \to (0, b)$ is a zero correlation linear hull of $\mathcal{E}_3$; and if the round functions are bijective, then for any $a \neq 0$, $(a, 0) \to (0, a)$ is a zero correlation linear hull of $\mathcal{E}_5$.

So far the longest integral distinguisher known for a Feistel structure with bijective round functions counts 4 rounds, and the longest integral distinguisher for a Feistel structure with general round functions counts 2 rounds. We improve these distinguishers by 1 round using Theorem 3.

Proposition 1. *Let $\mathcal{E}_r$ be an r-round Feistel structure defined on $\mathbb{F}_2^{2n}$. Then*

1. *If the F_i's are bijective, then for any $c \in \mathbb{F}_2^n$, $c \neq 0$, $c \cdot R_5$ is balanced on $\{(0,0),(c,0)\}^{\perp}$ with respect to $\mathcal{E}_5$.*
2. *If the F_i's are not necessarily bijective, then let $\{\alpha_0, \ldots, \alpha_{n-1}\}$ be a base of $\mathbb{F}_2^n$ over $\mathbb{F}_2$. Then $\alpha_{n-1} \cdot R_3$ is balanced on $\{(0, \sum_{i=0}^{n-2} c_i\alpha_i) | c_i \in \mathbb{F}_2\}^{\perp}$ with respect to $\mathcal{E}_3$.*

As a matter of fact, for any $c \in \mathbb{F}_2^n$, $c \neq 0$, $(c, 0) \to (0, c)$ is a zero correlation linear hull of $\mathcal{E}_5$. Thus according to Theorem 3, we can construct an integral distinguisher of $\mathcal{E}_5$, i.e., let (L_0, R_0) take all values in $\{(0,0),(c,0)\}^{\perp}$, then $c \cdot R_5$ is balanced.

6.2 24-Round Integral for CAST-256

The block cipher CAST-256 was proposed as a first-round AES candidate, and we refer to [34] for details. Firstly, we recall the following zero correlation linear property given in [17].

Property 1. $(0, 0, 0, L_1) \to (0, 0, 0, L_2)$ is a zero correlation linear hull of the 24-round CAST-256 (from the 13-th round to the 36-th round of CAST-256), where $L_1 \neq 0$, $L_2 \neq 0$ and $L_1 \neq L_2$.

Let $L_1^* = \{(l_1, l_2, \ldots, l_{31}, 0) | l_i \in \mathbb{F}_2\}$ and $L_2 = (0, \ldots, 0, 1)$. Then we obtain a zero correlation linear hull $(0, 0, 0, L_1^*) \to (0, 0, 0, L_2)$ for the 24-round CAST-256. According to Theorem 3, we can get the following result:

Proposition 2. *Let $V = \{(x_1, x_2, x_3, 0^{31}y) | x_i \in \mathbb{F}_2^{32}, y \in \mathbb{F}_2\}$. If the input takes all values in V, and let the output of the 24-round be $(C_0, C_1, C_2, C_3) \in \mathbb{F}_2^{32\times 4}$ (from the 13-th round to 36-th round). Then $(0, \ldots, 0, 1) \cdot C_3$ is balanced.*

Based on this integral distinguisher, we present a key recovery attack on 28-round CAST-256 which is the best known attack on CAST-256 in the non-weak key model. The details of the attack are listed the full version of this paper[31].

6.3 12-Round Integral for SMS4

The SMS4[35] block cipher is designed by the Chinese government as part of their WAPI standard for wireless networks. Up to date, the longest known integral distinguisher of SMS4 covers 10 rounds [36]. The details of SMS4 and the proof of the following propositions are listed in the full version of this paper[31].

Proposition 3. *Let* $V = \{v \in (\mathbb{F}_2^8)^4 | HW(v\mathcal{L}^T) = 1\}$*, where* $HW(x_1, x_2, x_3, x_4) = \#\{x_i \neq 0, i = 1, 2, 3, 4\}$*. For any* $d \in V$*,* $(0, 0, 0, d) \rightarrow (d, 0, 0, 0)$ *is a* 12*-round zero correlation linear hull of SMS4.*

Proposition 4. *Let* $V = \{v \in (\mathbb{F}_2^8)^4 | HW(v\mathcal{L}^T) = 1\}$*,* $V_d = \{w \in (\mathbb{F}_2^{32})^4 | (0, 0, 0, d) \cdot w = 0\}$*, and let* (c_0, c_1, c_2, c_3) *be the output of* 12*-round SMS4. Then for any* $d \in V$*, when the input takes all possible values in* V_d*, we have*

$$\#\{d \cdot c_0 = 0\} = \#\{d \cdot c_0 = 1\}.$$

Note that most of the known integral distinguishers are independent of the choices of the S-boxes. However, the integral distinguisher presented above is highly related with the S-boxes, since for different S-boxes, we would find different zero correlation linear hulls which lead to different integral distinguishers of SMS4.

6.4 8-Round Integral for Camellia without FL/FL^{-1} Layer

Based on the 8-round zero correlation linear hull presented in Example 1, we get the following 8-round integral of Camellia without FL/FL^{-1} layer:

Proposition 5. *Let* V *be defined as*

$$V = \{((x_1, \ldots, x_8), (x_9, \ldots, x_{16})) | x_1 \oplus x_2 \oplus x_5 \oplus x_7 \oplus x_8 = 0, x_i \in \mathbb{F}_2^8\}.$$

For any $h \in \mathbb{F}_2^8$*,* $h \neq 0$*,* $(h, 0, 0, h, 0, h, h, h) \cdot R_{i+8}$ *is balanced on* V *with respect to* 8*-round Camellia without* FL/FL^{-1} *layer.*

7 Conclusion

In this paper, we have investigated the link between impossible differential and integral cryptanalysis. To do this, we have introduced the concept of *structure* $\mathcal{E}$ and *dual structure* $\mathcal{E}^\perp$ and established the link in the following steps:

– We derived the relation between impossible differential of $\mathcal{E}$ and zero correlation linear hull of $\mathcal{E}^\perp$. We have shown that for a Feistel structure $\mathcal{F}_{SP}$ with SP-type round functions where P is invertible, constructing a zero correlation linear hull of $\mathcal{F}_{SP}$ is equivalent to constructing an impossible differential of $\mathcal{F}_{SP^T}$, which is the same structure as $\mathcal{F}_{SP}$ with P^T instead of P. For an SPN structure $\mathcal{E}_{SP}$, constructing a zero correlation linear hull of $\mathcal{E}_{SP}$ is equivalent to constructing an impossible differential of $\mathcal{E}_{S(P^{-1})^T}$, which is the same structure as $\mathcal{E}_{SP}$ with $(P^{-1})^T$ instead of P.

– We presented the relation between zero correlation linear hull and integral distinguisher of block ciphers. As proven in Sec. 4, a zero correlation linear hull always implies the existence of an integral distinguisher, while such statement only holds under certain conditions in [17]. Meanwhile, we have observed that the statement "*integral unconditionally implies zero correlation linear hull*" in [17] is correct only under the definition that integral property is a balanced vectorial boolean function, while it does not hold for the general case (i.e., integral defined in [3] is a zero-sum property).
– We built the link between impossible differential of $\mathcal{E}$ and integral distinguisher of $\mathcal{E}^{\perp}$. We have demonstrated that an r-round impossible differential of $\mathcal{E}$ always leads to an r-round integral distinguisher of $\mathcal{E}^{\perp}$. In the case that $\mathcal{E}$ and $\mathcal{E}^{\perp}$ are linearly equivalent, we obtained some direct links between impossible differential and integral distinguisher of $\mathcal{E}$. Specifically, an r-round impossible differential of an SPN structure, which adopts bit permutation as the linear layer, always indicates the existence of an r-round integral distinguisher.

The results and links presented in this paper not only allow to achieve a better understanding and classifying of impossible differential cryptanalysis, integral cryptanalysis and zero correlation linear cryptanalysis, but also provide some new insights with respect to these cryptanalytic approaches as shown below:

– The automatic search tool presented by Wu and Wang in Indocrypt 2012 finds all impossible differentials of both Feistel structures with SP-type round functions and SPN structures, which is useful in provable security of block ciphers against impossible differential cryptanalysis.
– Our statement "*zero correlation linear hull always implies the existence of an integral distinguisher*" provides a novel way for constructing integral distinguisher of block ciphers. With this observation, we have improved the integral of Feistel structures by 1 round, built a 24-round integral of CAST-256, proposed a 12-round integral of SMS4 which is 2-round longer than previously best known ones, and present an 8-round integral of Camellia without FL/FL^{-1} layers. These distinguishers could not be obtained by either the previously known methods for constructing integral distinguishers or by using the link given in [17]. Moreover, we have presented the best known key recovery attack on CAST-256 in non-weak key model to show that the new links can also be used to improve cryptanalytic results of some concrete ciphers.

By using the matrix representation given in [37], the concept of dual structure can be extended to generalized Feistel structures, and we can get similar results for these structures. Furthermore, we have focused on the links among the distinguishers used in impossible differential, integral and zero correlation linear cryptanalysis since distinguishers are the essential points in the evaluation of security margins of a block cipher against various cryptanalytic tools, and our results can be helpful in designing a block cipher from this point of view.

References

1. L.R. Knudsen. DEAL — A 128-bit Block Cipher. Department of Informatics, University of Bergen, Norway. Technical report, 1998.
2. E. Biham, A. Biryukov, A. Shamir. Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. EUROCRYPT 1999, LNCS 1592, pp. 12–23, Springer-Verlag, 1999.
3. L.R. Knudsen, D. Wagner. Integral Cryptanalysis. FSE 2002, LNCS 2365, pp. 112–127, Springer–Verlag, 2002.
4. A. Bogdanov, V. Rijmen. Linear Hulls with Correlation Zero and Linear Cryptanalysis of Block Ciphers. Designs, Codes and Cryptography, 70(3), pp. 369–383, 2014.
5. J. Kim, S. Hong, J. Lim. Impossible Differential Cryptanalysis Using Matrix Method. Discrete Mathematics, 310(5), pp. 988–1002, 2010.
6. Y. Luo, X. Lai, Z. Wu, G. Gong. A Unified Method for Finding Impossible Differentials of Block Cipher Structures. Information Sciences, Volume 263, 1 April 2014, Pages 211–220.
7. S. Wu, M. Wang. Automatic Search of Truncated Impossible Differentials for Word-Oriented Block Ciphers. Indocrypt 2012, LNCS 7668, pp. 283–302, 2012.
8. J. Daemen, L. R. Knudsen, V. Rijmen. The Block Cipher Square. Fast Software Encryption 1997, LNCS 1267, pp. 149–165, Springer–Verlag, 1997.
9. S. Lucks. The Saturation Attack — A Bait for Twofish. Fast Software Encryption 2001, LNCS 2355, pp. 1–15, Springer–Verlag, 2002.
10. A. Biryukov, A. Shamir. Structural Cryptanalysis of SASAS. EUROCRYPT 2001, LNCS 2045, pp. 394–405, Springer–Verlag, 2001.
11. X. Lai. Higher Order Derivatives and Differential Cryptanalysis. Communications and Cryptography: Two Sides of One Tapestry, 227 (1994)
12. L.R. Knudsen. Truncated and Higher Order Differentials. Fast Software Encryption 1994, LNCS 1008, pp. 196–211. Springer, Heidelberg (1995)
13. S. Picek, L. Batina, D. Jakobović, B. Ege, M. Golub. S-box, SET, Match: A Toolbox for S-box Analysis. WISTP 2014, LNCS 8501, pp. 140–149, 2014.
14. F. Chabaud, S. Vaudenay. Links Between Differential and Linear Cryptoanalysis. EUROCRYPT 1994, LNCS 950, pp. 356–365, Springer-Verlag, 1995.
15. B. Sun, R. Li, L. Qu, C. Li. SQUARE Attack on Block Ciphers with Low Algebraic Degree. Science China Information Sciences 53(10), pp. 1988–1995, 2010.
16. G. Leander. On Linear Hulls, Statistical Saturation Attacks, PRESENT and a Cryptanalysis of PUFFIN. EUROCRYPT 2011, LNCS 6632, pp. 303–322, Springer-Verlag, 2011.
17. A. Bogdanov, G. Leander, K. Nyberg and M. Wang. Integral and Multidimensional Linear Distinguishers with Correlation Zero. ASIACRYPT 2012, LNCS 7658, pp. 244–261, Springer–Verlag, 2012.
18. C. Blondeau and K. Nyberg. New Links Between Differential and Linear Cryptanalysis. EUROCRYPT 2013, LNCS 7881, pp. 388–404, Springer–Verlag, 2013.
19. C. Blondeau, G. Leander, K. Nyberg. Differential-Linear Cryptanalysis Revisited. FSE 2014, LNCS 8540, pp. 411–430, Springer–Verlag, 2015.
20. C. Blondeau, A. Bogdanov, M. Wang. On the (In)Equivalence of Impossible Differential and zero correlation Distinguishers for Feistel- and Skipjack-type Ciphers. ACNS 2014, LNCS 8479, pp. 271–288, 2014.
21. C. Blondeau, K. Nyberg. Links Between Truncated Differential and Multidimensional Linear Properties of Block Ciphers and Underlying Attack Complexities. EUROCRYPT 2014, LNCS 8441, pp. 165–182, 2014.

22. C. Carlet. Boolean Functions for Cryptography and Error Correcting Codes. Cambridge University Press, 2006.
23. A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, C. Vikkelsoe: PRESENT: An Ultra-Lightweight Block Cipher. CHES 2007, LNCS 4727, pp 450–466, 2007.
24. J. Borghoff, A. Canteaut, T. Güneysu, E. B. Kavun, M. Knezevic, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, P. Rombouts, S. Thomsen, T. Yalçın. PRINCE — A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract. ASIACRYPT 2012. LNCS 7658, pp. 208–225, 2012.
25. K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima and T. Tokita. Camellia: A 128–Bit Block Cipher Suitable for Multiple Platforms - Design and Analysis. SAC 2000, LNCS 2012, pp. 39–56, Springer–Verlag, 2000.
26. D. Kwon, J. Kim, S. Park, S.H. Sung etc. New Block Cipher: ARIA. ICISC 2003, LNCS 2971, pp.432–445, Springer-Verlag 2004.
27. H. Mala, M. Dakhilalian, V. Rijmen, M. Modarres-Hashemi. Improved Impossible Differential Cryptanalysis of 7-Round AES-128. INDOCRYPT 2010, LNCS 6498, pp. 282–291, Springer–Verlag, 2010.
28. W. Wu, W. Zhang, D. Feng. Impossible Differential Cryptanalysis of Round-Reduced ARIA and Camellia. Journal of Computer Science and Technology, 22(3), pp. 449–456, 2007.
29. A. Bogdanov, H. Geng, M. Wang, L. Wen, B. Collard. Zero Correlation Linear Cryptanalysis with FFT and Improved Attacks on ISO Standards Camellia and CLEFIA. SAC 2013, LNCS 8282, pp. 306–323.
30. L. Duo, C. Li, K. Feng. New Observation on Camellia. SAC 2005, LNCS 3897, pp. 51–64, Springer–Verlag, 2006.
31. B. Sun, Z. Liu, V. Rijmen, R. Li, L. Cheng, Q. Wang, H. Alkhzaimi, C. Li. Links among Impossible Differential, Integral and Zero Correlation Linear Cryptanalysis. http://eprint.iacr.org/2015/181.pdf.
32. C. Lee, Y. Cha. The Block Cipher: SNAKE with Provable Resistance against DC and LC Attacks. In Proceedings of 1997 Korea-Japan Joint Workshop on Information Security and Cryptology (JW–ISC'97), pp. 3–17, 1997.
33. S. Moriai, T. Shimoyama, T. Kaneko. Interpolation Attacks of the Block Cipher: SNAKE. FSE 1999, LNCS 1636, pp. 275–289, 1999.
34. First AES Candidate Conference. http://csrc.nist.gov/archive/aes/round1/conf1/aes1conf.htm.
35. Specification of SMS4, Block Cipher for WLAN Products — SMS4 (in Chinese), http://www.oscca.gov.cn/UpFile/200621016423197990.pdf
36. W. Zhang, B. Su, W. Wu, D. Feng. C. Wu. Extending Higher-Order Integral: An Efficient Unified Algorithm of Constructing Integral Distinguishers for Block Ciphers. ACNS 2012, LNCS 7341, pp. 117–134, Springer-Verlag, 2012.
37. T.P. Berger, M. Minier, G. Thomas. Extended Generalized Feistel Networks Using Matrix Representation. SAC 2013, LNCS 8282, pp. 289–305, 2014.

Chapter 13

Optimized Interpolation Attacks on LowMC

Publication Data

I. Dinur, Y. Liu, W. Meier, Q. Wang: Optimized Interpolation Attacks on LowMC. In Tetsu Iwata and Jung Hee Cheon (Eds.): *ASIACRYPT* 2015, Part II, volumn 9453 of *Lecture Notes in Computer Science*, pages 535-560, 2015.

Contributions

Major contributor of Section 4, 5 and 6, share 25% of workload.

Optimized Interpolation Attacks on LowMC

Itai Dinur[1], Yunwen Liu[2], Willi Meier[3], and Qingju Wang[2,4⋆]

[1] Département d'Informatique, École Normale Supérieure, Paris, France
[2] ESAT/COSIC, KU Leuven and iMinds, Belgium
[3] FHNW, Switzerland
[4] Department of Computer Science & Engineering, Shanghai Jiao Tong University, China

Abstract. LowMC is a collection of block cipher families introduced at Eurocrypt 2015 by Albrecht et al. Its design is optimized for instantiations of multi-party computation, fully homomorphic encryption, and zero-knowledge proofs. A unique feature of LowMC is that its internal affine layers are chosen at random, and thus each block cipher family contains a huge number of instances. The Eurocrypt paper proposed two specific block cipher families of LowMC, having 80-bit and 128-bit keys.

In this paper, we mount interpolation attacks (algebraic attacks introduced by Jakobsen and Knudsen) on LowMC, and show that a practically significant fraction of 2^{-38} of its 80-bit key instances could be broken 2^{23} times faster than exhaustive search. Moreover, essentially all instances that are claimed to provide 128-bit security could be broken about 1000 times faster. In order to obtain these results we optimize the interpolation attack using several new techniques. In particular, we present an algorithm that combines two main variants of the interpolation attack, and results in an attack which is more efficient than each one.

Keywords: Block cipher, LowMC, high-order differential cryptanalysis, interpolation attack.

1 Introduction

LowMC is a collection of block cipher families designed by Albrecht et al. and presented at Eurocrypt 2015. The cipher is specifically optimized for practical instantiations of multi-party computation, fully homomorphic encryption, and zero-knowledge proofs. In such applications, non-linear operations result in a heavy computational penalty compared to linear ones. The designers of LowMC took an extreme approach, combining very dense affine layers with simple non-linear layers that have algebraic degree of 2.

Perhaps the most distinctive feature of LowMC is that its affine layers are chosen at random, and thus each block cipher family contains a huge number

⋆ The fourth author is in part supported by the National Natural Science Foundation of China (no. 61472250), Major State Basic Research Development Program (973 Plan, no. 2013CB338004).

of instances. As this may enable a malicious party to instantiate LowMC with a hidden backdoor, its designers propose to use the Grain stream cipher [3] as a source of pseudo-random bits in order to restrict the freedom available in the LowMC instantiation. The designers also mention that it is possible to use any sufficiently random source to generate the affine layers, and this source does not necessarily need to be cryptographically secure.

The Eurocrypt paper proposed two specific block cipher families of LowMC, having 80-bit and 128-bit keys. The internal number of rounds in each family was set in order to guarantee a security level that corresponds to its key size. For this purpose, the resistance of LowMC was evaluated against a variety of well-known cryptanalytic attacks. One of the main considerations in setting the internal number of rounds was to provide resistance against algebraic attacks (such as high-order differential cryptanalysis [7]). Indeed, LowMC is potentially susceptible to algebraic attacks due to the low algebraic degree of its internal round, but the designers argue that LowMC has sufficiently many rounds to resist such attacks.

In this paper, we evaluate the resistance of LowMC against algebraic attacks and refute the designers' claims regarding its security level. Our results are given in Table 1, and show that a fraction of 2^{-38} of the LowMC 80-bit key instances could be broken in about 2^{57} time, using 2^{39} chosen plaintexts. The probability of 2^{-38} is practically significant, namely, a malicious party can easily find weak instances of LowMC by running its source of pseudo-random bits with sufficiently many seeds, and checking whether the resultant instance is weak (which can be done efficiently using basic linear algebra).

For LowMC with 128-bit keys, we describe an attack that breaks a fraction of 2^{-122} of its instances in time 2^{86} using 2^{70} chosen plaintexts. We note that this specific attack does not violate the formal security claims of the LowMC designers, as they do not consider attacks that apply to less than 2^{-100} of the instances as valid. Nevertheless, the designers of LowMC allow to instantiate it using a pseudo-random source that is not cryptographically secure. Our result shows that this is risky, as using an over-simplified source for pseudo-randomness may give a malicious party additional control over the LowMC instantiation, and allow finding weak instances much faster than exhaustively searching for them in 2^{122} time.

Finally, we describe an attack that can break essentially all LowMC instances with 128-bit keys. Although the attack is significantly slower than the weak-instance attack, it is still about 1000 times faster than exhaustive search, and uses 2^{73} chosen plaintexts.

All of our results were obtained using the interpolation attack, which is an algebraic attack introduced by Jakobsen and Knudsen in 1997 [4]. In an interpolation attack, the attacker considers some intermediate encryption value b as a polynomial in the ciphertext bits. The aim of the attacker is to interpolate the algebraic normal form (ANF) of b by recovering its unknown coefficients, and this typically allows to recover the secret key using ad-hoc techniques.

Instance Family	Number of Rounds	Section	Rounds Attacked	Fraction of Instances	Data†	Time††	Memory†††
LowMC-80	11	6.1	9	1	2^{35}	2^{38}	2^{35}
		6.2	10	1	2^{39}	2^{57}	2^{39}
		6.3	all (11)	2^{-38}	2^{39}	2^{57}	2^{39}
LowMC-128	12	7.1	11	1	2^{70}	2^{86}	2^{70}
		7.1	all (12)	2^{-122}	2^{70}	2^{86}	2^{70}
		7.2	all (12)	1	2^{73}	2^{118}	2^{80}

† Given in chosen plaintexts.
†† Given in LowMC encryptions.
††† Given in 256-bit words.

Table 1. Attacks on LowMC

In order to recover the unknown coefficients, the attacker allocates a variable for each one of them. Assuming that b has a low-degree representation in terms of the plaintext bits, the attacker collects linear equations on the variables, typically by using high-order differentials in a chosen plaintext attack. After obtaining sufficiently many equations, the unknown variables are recovered by solving the resultant linear equation system. The efficiency of the attack depends on the algebraic degree of b in terms of the plaintext, but also on the number of allocated variables which is determined by the number of unknown coefficients in the ANF representation of b in terms of the ciphertext.

Although our results were obtained using the well-known interpolation attack, its straightforward application does not seem to threaten the security of LowMC. Therefore, we had to develop new techniques such as using carefully chosen plaintext structures which allow to efficiently derive the linear system of equations. However, our main new contribution is described next by considering two variants of the interpolation attack.

In the original variant of the interpolation attack over $GF(2)$ (which we refer to as variant 1), the attacker views the ANF of some intermediate encryption bit b as an initially unknown polynomial $F_K(C)$ in the ciphertext bits $C = c_1, \ldots, c_n$, where $K = x_1, \ldots, x_\kappa$ is the unknown (fixed) secret key. In a dual approach to the interpolation attack, which we refer to as variant 2 (used, for example, in [8]), the attacker interpolates the full polynomial $F(K, C)$ by considering each monomial in the key bits $x_1, \ldots, x_\kappa$ with a non-zero coefficient as a separate (linearized) variable. For example, consider the polynomial

$$F(c_1, c_2, x_1, x_2, x_3) = c_1c_2x_1 + c_1c_2x_2 + c_1x_1 + c_1x_2 + c_2x_1 + x_1x_2 + x_3 + 1.$$

We can write

$$F_{(x_1,x_2,x_3)}(c_1, c_2) = \alpha_1c_1c_2 + \alpha_2c_1 + \alpha_3c_2 + \alpha_4,$$

and thus in the first variant we have 4 variables: $\alpha_1, \alpha_2, \alpha_3, \alpha_4$. In this variant, the actual representation of the variables in terms of the key is not considered.

In the dual variant, we write

$$F(c_1, c_2, x_1, x_2, x_3) = x_1x_2(1) + x_1(c_1c_2 + c_1 + c_2) + x_2(c_1c_2 + c_1) + x_3(1) + 1,$$

and we have 4 variables: x_1x_2, x_1, x_2, x_3.

The advantage of variant 2 over the first variant is that it directly recovers the secret key, and furthermore, in some cases it may result in a smaller number of variables in the equation system. At the same time, in order to derive the actual equation system the attacker has to evaluate the polynomial F for each ciphertext. This process is less efficient in variant 2, since each evaluation of $F(K, C)$ is expensive (it requires evaluating all the complex ciphertext expressions that are multiplied with the variables), whereas in variant 1 each evaluation of $F_K(C)$ is relatively simple (it requires evaluating simple monomials in the ciphertext). Therefore, the choice of which variant to use in order to optimize the attack depends on the underlying cryptosystem.

Our main idea is to combine the two dual variants of interpolations attacks: we first derive the equation system efficiently using the original variant of [4]. Then, we transform a carefully chosen variable subset to variables which are linearized monomials in the key bits, as in variant 2. This results in a mixed variable set that is smaller than the variable sets of each variant. Consequently, we obtain an attack which is more efficient than each one of the two variants.

In our example above, we can express $\alpha_1 = x_1 + x_2$, $\alpha_2 = x_1 + x_2$ and $\alpha_3 = x_1$, resulting in only 3 variables: x_1, x_2, α_4. Obviously, our toy example merely demonstrates the idea at a very high level, and the actual choice of which variables to transform as well as the analysis of the resultant algorithm are more involved.

The paper is organized as follows. In Section 2 we give some preliminaries, while in Section 3 we give a brief description of LowMC. Our basic attack on 9-round LowMC with an 80-bit key is described in Section 4, while our generic framework for optimized interpolation attacks is described in Section 5. In sections 6 and 7 we apply our optimized attack to LowMC with 80 and 128-bit keys, respectively. Finally, we conclude the paper in Section 8.

2 Preliminaries

In this section, we describe preliminaries that are used in the rest of the paper.

2.1 Boolean Algebra

For a finite set S, denote by $|S|$ its size. Given a vector $u = (u_1, \ldots, u_n) \in GF(2^n)$, let $wt(u)$ denote its Hamming weight.

Any function F from $GF(2^n)$ to $GF(2)$ can be described as a multivariate polynomial, whose algebraic normal form (ANF) is unique and given as $F(x_1, \ldots, x_n) = \sum_{u=(u_1,\ldots,u_n)\in GF(2^n)} \alpha_u M_u$, where $\alpha_u \in \{0, 1\}$ is the coefficient of

the monomial $M_u = \prod_{i=1}^{n} x_i^{u_i}$, and the sum is over $GF(2)$. The algebraic degree of the function F is defined as $deg(F) \triangleq max\{wt(u)|\alpha_u \neq 0\}$. Therefore, a function F with a degree bounded by $d \leq n$ can be described using $\sum_{i=0}^{d} \binom{n}{i}$ coefficients. To simplify our notations, we define $\binom{n}{\leq d} \triangleq \sum_{i=0}^{d} \binom{n}{i}$.

The ANF coefficient α_u of F can be interpolated by summing (over $GF(2)$) over $2^{wt(u)}$ evaluations of F: define the set of inputs S to contain all the $2^{wt(u)}$ n-bit vectors whose bits set to 1 is a subset of the bits set to 1 in $u_1, \ldots, u_n$. More formally, let $S = \{x = (x_1, \ldots, x_n)|\bar{u} \wedge x = 0\}$ (where $\bar{u}$ is bitwise NOT applied to u , and $\wedge$ is bitwise AND), then $\alpha_u = \sum_{(x_1,\ldots,x_n)\in S} F(x_1, \ldots, x_n)$. Note that this implies that a function F with a degree bounded by $d \leq n$ can be fully interpolated given its evaluations on the set of $\binom{n}{\leq d}$ inputs whose Hamming weight is at most d, namely $\{x = (x_1, \ldots, x_n)|wt(x) \leq d\}$.

Given the truth table of an arbitrary function F (as a bit vector of 2^n entries), the ANF of F can be represented as a bit vector of 2^n entries, corresponding to its 2^n coefficients α_u. This ANF representation can be efficiently computed using the *Moebius transform*, which is an FFT-like algorithm. The Moebius transform performs n iterations on its input vector (the truth table of F), where in each iteration, half of the array entries are XORed into the other half. In total, its complexity is about $n \cdot 2^n$ bit operations. For more details on the Moebius transform, refer to [5].

2.2 High-Order Differential Cryptanalysis and Interpolation Attacks

In this section, we give a brief summary of high-order differential cryptanalysis and interpolation attacks.

High-Order Differential Cryptanalysis High-order differential cryptanalysis was introduced in [7] as an algebraic attack that is particularly efficient against ciphers of low algebraic degree. The basic variant of high-order differential cryptanalysis over $GF(2)$ considers some target bit b (which can be either a ciphertext or an intermediate encryption value) and analyzes its ANF representation in terms of the plaintext P, denoted by $F_K(P)$ (where K is the unknown secret key). Given that $deg(F_K(P)) \leq dg$ independently of K for dg (relatively) small, then the attacker chooses an arbitrary linear subspace S of dimension $dg + 1$, and evaluates the cipher (in a chosen plaintext attack) over its 2^{dg+1} inputs. Since every differentiation reduces the algebraic degree of the target bit by 1 and $deg(F_K(P)) \leq dg$, the value of the high-order differential over S for the target bit b (namely, the sum of evaluations of b over $GF(2)$) is equal to zero (refer to [7] for details). High-order differential properties may be used in key recovery attacks, depending on the specification of the cipher (refer to [6]).

However, such key recovery methods are not part of the framework described in this section.

Interpolation Attacks The interpolation attack was introduced in 1997 by Jakobsen and Knudsen as an algebraic attack on block ciphers [4]. The attack is closely related to high-order differential cryptanalysis[1] and (similarly to high-order differential cryptanalysis) is particularly efficient against block ciphers whose round function is of low algebraic degree. The interpolation attack has several variants, and can be applied over a general finite field, exploiting known or chosen plaintexts. Here, we give a high-level description of the chosen plaintext interpolation attack over $GF(2)$, as this is the variant we apply to LowMC.

The attack considers some intermediate encryption target bit b of the block cipher, whose ANF representation can be expressed from the decryption side in terms of the ciphertext and key as $F(C, K)$. The key K is viewed as an unknown constant, and thus we can write $F_K(C) = F_K(c_1, \ldots, c_n) = \sum_{u=(u_1,\ldots,u_n)\in GF(2^n)} \alpha_u M_u$, where $\alpha_u \in \{0, 1\}$ is the coefficient of the monomial $M_u = \prod_{i=1}^{n} c_i^{u_i}$. Therefore, the coefficients α_u of $F_K(C)$ generally depend on the secret key and are unknown in advance. The goal of the interpolation attack is to recover (interpolate) the unknown coefficients of $F_K(C)$, and then use various ad-hoc techniques (which are not part of the framework described in this section) in order to recover the actual secret key.

In order to deduce the unknown coefficients of $F_K(C)$, they are considered as variables (i.e., linearized), and recovered by solving a linear equation system. For the purpose of constructing the equation system, the attacker assumes that the algebraic degree dg of the bit b in terms of the bits of the plaintext is relatively small, which allows to use high-order differential cryptanalysis (as described above). More specifically, a high-order differential property is devised by encrypting a subspace S of plaintexts of dimension $dg + 1$, and performing high-order differentiation with respect to this subspace, whose outcome is zero on the bit b.

When expressed in terms of the ciphertexts $C_1, \ldots, C_{2^{dg+1}}$ (obtained by encrypting the plaintexts of S), this gives the equation $\sum_{t=1}^{2^{d+1}} F_K(C_t) = 0$. For each ciphertext C_t, $F_K(C_t)$ is merely a linear expression in the variables α_u (the coefficient of α_u in this expression is easily deduced by evaluating M_u on C_t), and thus the subspace S gives rise to one linear equation in the variables α_u. In order to solve for the unknown variables α_u, the attacker considers several such subspaces, each giving one equation. In total, the number of equations (and subspaces considered) needs to be roughly equal to the number of the unknown α_u variables, assuming the equations are sufficiently "random".

[1] In fact, some of its variants directly exploit high-order differential properties, as we describe next.

From the high-level description above, it is easy to conclude that the data and time complexities of the attack depend on the value of the degree dg and the number of unknown variables α_u. Therefore, in order to mount efficient interpolation attacks, the attacker tries to minimize these parameters, as we demonstrate in our attacks on LowMC.

2.3 Model of Computation

Since an exhaustive key search attack (which evaluates the LowMC encryption function) and our attacks use different bitwise operations, comparing these attacks cannot be done simply by counting the number of encryption function evaluations. Instead, we compare the complexity of straight-line implementations of the algorithms, counting the number of bit operations (such as XOR, AND, OR) on pairs of bits. This computation model ignores operations such as moving a bit from one position to another (which only requires renaming variables in straight-line programs). As calculated in Section 3, the straight-line implementation of one encryption function evaluation of LowMC requires about 2^{19} bit operations. Consequently, a straight-line implementation of exhaustive search for 80-bit and 128-bit keys requires about 2^{99} and 2^{147} bit operations, respectively, and these are quantities of reference for our attacks.

3 Description of LowMC

LowMC is a collection of SP-network instances, proposed at Eurocrypt 2015 [1] by Albrecht et al. The specification defined two specific instance families which are analyzed in this paper, both having a block size of $n = 256$ bits, and are characterized by their key size κ, which is either 80 or 128 bits. In this paper, we refer to these instance families as LowMC-80 and LowMC-128. The encryption function of LowMC applies a sequence of rounds to the plaintext, where each round contains a (bitwise) round-key addition layer, an Sbox layer, and an affine layer (over $GF(2)$). LowMC was designed with distinct features (as detailed in the pseudocode below): it has a linear key schedule and its affine layers are selected at random, where each selection defines a separate instance of the family. The Sbox layer of LowMC is composed of 3-bit Sboxes with degree 2 over $GF(2)$ (the actual specification of the Sboxes is irrelevant for our analysis and is omitted from this paper). Furthermore, the Sbox layers are only partial, namely, in each Sbox layer, only $3m < n$ bits go through an Sbox (where m is a parameter), while the rest of the $n - 3m$ bits remain unchanged.

Each family instance of LowMC is also defined with a data limit lim, which determines the maximal (recommended) data complexity before changing the key. In other words, the cipher is guaranteed to offer security according to its key size as long as the adversary cannot obtain more than 2^{lim} plaintext-ciphertext pairs. The parameters of the two instance families are given in Table 2.

The pseudocode of the encryption function (taken from [1]) is given below.

Instance Family	key size κ	Block Size n	Sboxes m	Data lim	Rounds r
LowMC-80	80	256	49	64	11
LowMC-128	128	256	63	128	12

Table 2. LowMC Instance Families

```
ciphertext = encrypt (plaintext,key)
  //initial whitening
  state = plaintext + MultiplyWithGF2Matrix(KMatrix(0),key)
  for (i = 1 to r)
    //m computations of 3-bit Sbox, n-3m bits remain the same
    state = Sboxlayer (state)
    //affine layer
    state = MultiplyWithGF2Matrix(LMatrix(i),state)
    state = state + Constants(i)
    //generate round key and add to the state
    state = state + MultiplyWithGF2Matrix(KMatrix(i),state)
  end
  ciphertext = state
```

The matrices $LMatrix(i)$ are chosen at random from all invertible binary $n \times n$ matrices, while the matrices $KMatrix(i)$ are chosen independently and uniformly at random from all binary $n \times \kappa$ matrices of rank $min(n, \kappa)$. The constants $Constants(i)$ are chosen independently and uniformly at random from all binary vectors of length n.

In this paper, we denote the 256-bit state at the input to the i'th key addition layer by X_{i-1} (e.g., the plaintext is denoted X_0), the input to the i'th Sbox layer by Y_{i-1} and the input to the i'th affine layer by Z_{i-1}. We refer to the $3m$ bits of the state that go through Sboxes in the Sbox layer as the S-part, while the remaining $n - 3m$ bits are referred to as the I-part. Given a state W, denote by $W|SP$ and $W|IP$ the S-part and I-parts of the state, respectively (e.g., $Y_5|IP$ is the I-part of the input state to the 6'th Sbox layer).

It is common practice in cryptanalysis of block ciphers to exchange the order of the final two affine operations over $GF(2)$ (namely, the keyless affine transformation and key addition). This allows the attacker to "peel off" the last affine transformation at a negligible cost by working with an equivalent last-round key (obtained by an affine transformation on the original last-round key). For the sake of simplicity, we assume in the following that we have already "peeled off" the last affine transformation of the cipher. Therefore, the final states of the last round r are denoted by X_{r-1}, Y_{r-1}, Z_{r-1} and Y_r, which denotes the ciphertext (after "peeling off" the final affine transformation).

Each affine layer of LowMC involves multiplication of the 256 state with a 256×256 matrix. This multiplication requires roughly 2^{16} bit operations, and therefore a single encryption of LowMC (that contains more than 8 rounds) requires more than $2^{16} \cdot 8 = 2^{19}$ bit operations (as already noted in Section 2.3).

4 A Basic 9-Round Attack on LowMC-80

In this section we describe our basic interpolation attack on 9-round LowMC, which is given first without optimizations for the sake of clarity. We begin by considering the elements that are required for the attack.

4.1 The High-Order Differential Property

We construct the high-order differential property used in the interpolation attack. A similar property was described by the LowMC designers [1], but we reiterate it here for the sake of completeness.

The algebraic degree of a single round of LowMC-80 over $GF(2)$ is 2, and therefore the algebraic degree of any bit at the input to the 6'th Sbox layer of LowMC-80, Y_5, in the input bits, X_0, is at most 32. Moreover, as the bits of the I-part of LowMC do not go through Sboxes in the first round, then the degree at the input to the 7'th Sbox layer, Y_6, in the bits of the I-part, $X_0|IP$, (given that the input bits of the S-part, $X_0|SP$, are constant) is at most 32. Furthermore, since the bits of the I-part of the 7'th Sbox layer do not go through an Sbox, the degree of any bit of $Z_6|IP$ in the input bits of the I-part, $X_0|IP$, is at most 32 (given that $X_0|SP$ is constant).

The last property implies that the value of a 33-order differential over any 33-dimensional subspace selected from $X_0|IP$, (keeping $X_0|SP$ constant) is zero for any bit of $Z_6|IP$. Moreover, as we selected a subspace whose bits do not go through an Sbox in the first round, the value of a 32-order differential for any bit of $Z_6|IP$ over any 32-dimensional subspace from $X_0|IP$, is a constant (independent of the key). This observation implies that we can select several 32-dimensional subspaces, and compute in a preprocessing phase the constants obtained by summing (over $GF(2)$) over a target bit of $Z_6|IP$ (for an arbitrary fixed value of the key). Each such constant (derived from a 32-dimensional subspace) gives one bit of information that we will exploit as the constant value of an equation in the interpolation attack.

4.2 Bounding the Number of Variables

In the interpolation attack on 9-round LowMC-80, we select a target bit from $Z_6|IP$ and denote its ANF representation in the 256-bit ciphertext (obtained after inverting the final affine transformation) and 80-bit key by $F(C, K)$. We consider K as an unknown constant, and write $F_K(C) = F_K(c_1, \ldots, c_{256}) = \sum_{u=(u_1,\ldots,u_{256})\in GF(2^{256})} \alpha_u M_u$, where $\alpha_u \in \{0, 1\}$ is the coefficient of the monomial $M_u = \prod_{i=1}^{256} c_i^{u_i}$. As the complexity of the attack depends on the number of variables α_u, it is important to estimate their number with good accuracy. An initial estimation can be made by observing that the algebraic degree of the (inverse) round of LowMC-80 is 2,[2] and thus $deg(F_K(C)) \leq 4$. This implies that $\alpha_u = 0$

[2] The algebraic degree of any invertible 3-bit Sbox is (at most) 2.

in case $wt(u) > 4$, and therefore the number of unknown variables is upper bounded by $\binom{256}{\leq 4} \approx 2^{27}$.

The initial upper bound on the number of variables can be significantly improved by considering the specific round function of LowMC-80. For this purpose, it will be convenient to use additional notation to describe the variables α_u according to the degree of M_u, by defining the set of variables U_i for a positive integer i as $U_i = \{\alpha_u \text{ that is not identically zero as a function of the key} | wt(u) = i \bigwedge u \in GF(2^{256})\}$. We have already seen that U_i is empty for $i > 4$ (as these variables are identically zero independently of the key), and we now derive tighter bounds on $|U_i|$ for $i \leq 4$. Thus, we analyze the symbolic representation of the state variables in the decryption direction, starting from the ciphertext Y_9, up to Z_6, as polynomials in the ciphertext bits $c_1, \ldots, c_{256}$.

The ciphertext Y_9 contains 256 bits of $c_1, \ldots, c_{256}$, while in order to compute Z_8 we merely add (unknown) constants to these bits (recall that we "peeled off" the last affine layer). Then, the inverse Sbox layer is applied to Z_8 to obtain the state Y_8. Each 3-bit Sbox may contribute (up to) 3 quadratic monomials to Y_8, and 6 monomials in total, e.g., an Sbox corresponding to ciphertext bits c_1, c_2, c_3 may contribute the monomials $c_1, c_2, c_3, c_1c_2, c_1c_3, c_2c_3$. Note that these monomials may appear in the ANF of different bits of Y_8 with different unknown coefficients (e.g., c_1x_1 and c_1x_2 may appear in the ANF of two different bits of Y_8). However, in interpolation attacks, we consider the ANF of the target bit, in which the coefficient α_u of every monomial M_u in the ciphertext is linearized and considered as a single variable. Therefore, the important quantity is the number of possibilities to create the monomials M_u (for this reason, the monomial c_1 is counted only once even if it appears in the ANF of different bits of Y_8 with different unknown coefficients).

Since there are 49 Sboxes, the total number of monomials M_u in the ANF of the state bits of Y_8 is bounded by $|U_2| \leq 3 \cdot 49 = 147$, $|U_1| \leq 256$ (which is the trivial bound) and $|U_i| = 0$ for $i \geq 3$. As the affine and key addition mappings do not influence the number of monomials M_u, this bound applies also to X_8 and Z_7.

Next, the inverse Sbox layer is applied to Z_7 to obtain the state Y_7, for which we already know that $|U_i| = 0$ for $i > 4$. Since the Sbox layer is of degree 2, a trivial upper bound on the number of variables α_u in Y_7 is obtained by multiplying the $147 + 256 = 403$ monomials in unordered pairs, giving $|\bigcup_{i=1}^{4} U_i| \leq \binom{403}{2} + 403 < 2^{16.5}$. Since the key addition and affine layers do not influence the number of monomials, the upper bound of $2^{16.5}$ also applies to X_7 and Z_6, and it is much smaller than our initial bound of about 2^{27}.

We denote the set of variables $\bigcup_{i=1}^{4} U_i$ by U, and note that the explicit set $\{u | \alpha_u \in U\}$ (which gives the relevant monomials M_u) can be easily derived during preprocessing (which involves a more explicit computation of the monomial set $\{M_u | \alpha_u \in U\}$, whose size is bounded above).

4.3 Obtaining the Data

After deducing that the number of variables in the system of equations is $|U| \approx 2^{16.5}$, we conclude that we need to differentiate over about $2^{16.5}$ 32-dimensional subspaces in order to obtain sufficiently many equations to solve the system. A trivial way to do this is to select about $2^{16.5}$ arbitrary linearly independent 32-dimensional subspaces from the $256 - 3 \cdot 49 = 109$ bits of $X_0|IP$. This results in an attack with data complexity of $2^{32+16.5} = 2^{48.5}$, and is rather wasteful. A more efficient approach (which was previously used in various papers such as [2]), is to select a large 37-dimensional subspace S from $X_0|IP$, containing $\binom{37}{32} > 2^{18}$ linearly independent 32-dimensional subspaces, which should suffice for the attack (assuming that the constructed system of equations is sufficiently random). The subspaces are indexed according to $37 - 32 = 5$ constant indexes that are set to zero in S.

4.4 The Basic Interpolation Attack

We now describe a basic interpolation attack on 9-round LowMC-80. We note that this attack is incomplete, as it only computes the $|U|$ variables α_u using $e \approx |U|$ equations, without recovering the actual secret key. The details of this final step will be given in the optimized attack in Section 5.2. For the sake of convenience, we describe the attack in two phases: the preprocessing phase (which is independent of the data and secret key) and online phase. However, we take into account both phases in the total complexity evaluation.

Assume we selected a target bit b from $Z_6|IP$, a subspace S of dimension 37 from $X_0|IP$, and $e \approx |U|$ 32-dimensional subspaces $S_1, \ldots, S_e$ in S. The detailed attack is described below.

Preprocessing:

1. Compute an e-bit array of free coefficients for $e \approx |U|$ equations, denoted by $\boldsymbol{a_0}$: evaluate b on the subset of inputs of S (with the key set to zero), and obtain a bit array of size 2^{37}. Finally, calculate the free coefficients by summing on b for the e 32-dimensional subspaces $S_1, \ldots, S_e$ in S, and store the result in $\boldsymbol{a_0}$.
2. Calculate the $|U|$ vectors $\{u|\alpha_u \in U\}$: This can be done by first calculating the 403 monomials M_u past the first Sbox layer, and multiplying them in pairs (as described in Section 4.2).

Online:

1. Ask for the encryptions of the 2^{37} plaintexts in S and store the ciphertexts in a table.
2. Allocate a $2^{37} \times |U|$ matrix A, where row $A[t]$ is a bit array that represents the evaluation $F_K(C_t)$ (namely, $\sum_{\{u|\alpha_u \in U\}} \alpha_u M_u(C_t)$).

3. For each ciphertext C_t, calculate $A[t]$ by evaluating $F_K(C_t)$:
   (a) For each $\{u|\alpha_u \in U\}$, evaluate the monomial $M_u(C_t)$ (the coefficient of α_u) and set the corresponding bit entry in $A[t]$ according to the result.
4. Allocate an $e \times |U|$ matrix E over $GF(2)$, representing the equation system on U.
5. For each 32-dimensional subspace S_j in S, namely $S_1, \ldots, S_e$ (that match the subspaces considered in preprocessing Step 1):
   (a) Populate the row (equation) $E[j]$ by summing over the 2^{32} rows of A corresponding to S_j.
6. Solve the equation system $E\boldsymbol{x} = \boldsymbol{a_0}$, where $\boldsymbol{x}$ represents the vector of variables of U and $\boldsymbol{a_0}$ is the vector of free coefficients calculated in preprocessing Step 1.

The data complexity of the attack is 2^{37} chosen plaintexts. The total time complexity of the attack is about 2^{65} bit operations, dominated by online Step 5 (for each of the e subspaces, we sum over 2^{32} bit vectors of size $|U|$, requiring about $e \cdot 2^{32} \cdot |U| \approx 2^{65}$ bit operations). The memory complexity of the attack is about $2^{37} \cdot |U| \approx 2^{53.5}$ bits, dominated by the storage of the matrix A in online Step 2.

We note that in the complexity evaluation of the attack we ignore indexing issues that arise (for example) in Step 3.a (that maps between a variable $\alpha_u \in U$ and its corresponding column index in $A[t]$), and in Step 5 (that maps between a subspace S_j in S and the corresponding 5 constant indexes of S). The reason that we can ignore these mappings in the complexity evaluation is that they are independent of the secret key and data, and therefore, they can be precomputed and integrated into the straight-line implementation of the program.

5 The Optimized Interpolation Attack

In this section, we introduce three optimizations of the basic 9-round attack above. The first optimization reorders the steps of the algorithm in order to reduce the memory complexity, while the second optimization further exploits the structure of chosen plaintexts to reduce the time complexity of the attack. Finally the third optimization is based on a novel technique in interpolation attacks, and allows to (further) reduce the data and time complexities. We first describe informally how to apply the optimizations to the basic 9-round attack on LowMC-80 above, and then devise a more formal and generic framework that can be applied to other LowMC variants.

The first two optimizations focus on online steps 2–5, which compute the equation system E from the 2^{37} ciphertexts. First, we reduce the memory complexity by noticing that we do not need to allocate the matrix A. Instead, we work column-wise and focus on a single column $A[*][\ell]$ at a time, corresponding to some $\{u|\alpha_u \in U\}$. We evaluate $M_u(C_t)$ for all ciphertexts (which gives an array of 2^{37} bits, $\boldsymbol{a_\ell}$) and then populate the corresponding column $E[*][\ell]$ by summing over the 32-dimensional subspaces $S_1, \ldots, S_e$ on $\boldsymbol{a_\ell}$.

Next, we reduce the time complexity by optimizing the summation process: given a bit array $\boldsymbol{a_\ell}$ of 2^{37} entries, the goal is to sum over many 32-dimensional subspaces (indexed according to 5 bits which are set to zero). This can be done efficiently using the Moebius transform (refer to Section 2.1). For this purpose, we can view $\boldsymbol{a_\ell}$ as evaluating a 37-variable polynomial over $GF(2)$, and the summation over a 32-dimensional subspace of $\boldsymbol{a_\ell}$ is equal to the coefficient of its corresponding 32-degree monomial. All these coefficients are computed by the Moebius transform in about $37 \cdot 2^{37}$ bit operations. We stress that the reason that we can use the Moebius transform in this case is purely combinatorial and is due to the way that we selected the structure of subspaces for the interpolation attack. Indeed, there does not seem to be any obvious algebraic interpretation to $\boldsymbol{a_\ell}$ when viewed as a polynomial.

Finally, we optimize the data complexity (and further reduce the time complexity): In order to achieve this, examine the polynomial $F(K, C)$ (as a function of both the key and ciphertext) for the target bit b selected in $Z_6|IP$. Due to the linear key schedule of LowMC, this polynomial is of degree 4, similarly to $F_K(C)$ (in which the key is treated as a constant). We consider a variable $\alpha_u \in U$ and analyze its ANF in terms of the 80 key bit variables. Since α_u is multiplied with M_u in $F(K, C)$, then $deg(\alpha_u) + deg(M_u) \leq 4$, implying that if $deg(M_u) \geq 2$, then $deg(\alpha_u) \leq 2$. This simple observation is borrowed from cube attacks [2] and can be used to significantly reduce the number of variables U, as described next.

Consider all the variables in $U_2 \bigcup U_3 \bigcup U_4$, and recall that their number was upper-bounded in Section 4.2 by roughly $2^{16.5}$. However, since all of these variables are polynomials of degree (at most) 2 in the 80 key bits, they reside in a linear subspace of monomials of dimension $\binom{80}{2} + 80 = 3240$. This implies that we can significantly reduce the total number of variables from $\approx 2^{16.5}$ to $3240 + 256 = 3496 < 2^{12}$ (including the 256 variables of U_1) by considering linear relations between the variables $U_2 \bigcup U_3 \bigcup U_4$. An immediate consequence of the reduction of variables is that we need less equations to solve the equation system, and therefore, we require less subspaces (or data) to obtain these equations. More specifically, a subspace of dimension 35 contains $\binom{35}{32} = 6545 > 2^{12}$ subspaces of dimension 32, which should suffice for the attack.

Assuming that we interpolate the variables of $U_2 \bigcup U_3 \bigcup U_4$ in terms of the key and recover their values, then the key itself should be very easy to deduce, as the variables of U_3 are merely key bits.

We note that while the idea above exploits the linear key schedule of LowMC, the technique is general and can be applied to block ciphers with arbitrary key schedules. In this case, it would consider each round key as independent. This increases the number of variables in the (linearized) key, but not necessarily by a significant factor. For example, if LowMC-80 had a non-linear key schedule, the optimization above would interpolate $U_2 \bigcup U_3 \bigcup U_4$ in terms of $\binom{80}{2} + 80 = 3240$ monomials in the key of round 9, and only 80 additional linear monomials and $3 \cdot 49 = 294$ quadratic monomials in the key of round 8 that are created by the inverse Sbox layer of round 8 (we can assume that the key of round 8 is

added right after the 8'th Sbox layer, as the key addition and affine layer are interchangeable).

5.1 Transformation of Variables

In this section, we begin to describe our generic framework for interpolation attacks on LowMC by formalizing the last optimization described above.

Given an instance of LowMC with a 256-bit block, a key size of κ, and m Sboxes per layer, we assume that we want to interpolate a target bit b through the final r_1 rounds of the cipher. We first describe in a more generic way how to calculate the initial set of variables U, and bound its size. As in the 9-round attack, the number of monomials in the 256 ciphertext bits at Y_{r-1} (after inverting the final Sbox layer) is bounded by $256 + 3m$. The target bit b is a polynomial of degree 2^{r_1-1} in the state Y_{r-1}, and thus it contains at most $\binom{256+3m}{\leq 2^{r_1-1}}$ monomials. Therefore, the set of monomials with (apriori) unknown coefficients can be computed by multiplying the $256 + 3m$ monomials in unordered tuples (with no repetition) of size up to 2^{r_1-1}. Thus,

$$|U| \leq \binom{256 + 3m}{\leq 2^{r_1-1}},$$

and this set can be computed with $|U|$ multiplications of tuples. Note again that this bound is generally better than the trivial bound of $|U| \leq \binom{256}{\leq 2^{r_1}}$, which is obtained due to the fact that b is a polynomial of degree 2^{r_1} in the 256 ciphertext bits.

We consider the target bit b as a polynomial in both the ciphertext and the key, namely, $F(K, C) = F(x_1, \ldots, x_\kappa, c_1, \ldots, c_{256}) = \sum_{u=(u_1,\ldots,u_n)\in GF(2^n)} \alpha_u M_u$, where $M_u = \prod_{i=1}^{n} c_i^{u_i}$ and $\alpha_u(x_1, \ldots, x_\kappa)$ is a polynomial from $GF(2^\kappa)$ to $GF(2)$. We partition the variables of $|U|$ into subsets according to the degree of their monomials in the ciphertext, which is bounded by $deg(F_K(C)) = 2^{r_1}$. Denote $d = 2^{r_1}$ and write $U = \bigcup_{i=1}^{d} U_i$, where $U_i = \{\alpha_u \in U | deg(M_u) = i\}$. Due to the linear key schedule of LowMC, we have $deg(F(K, C)) = deg(F_K(C)) = d$, and therefore $deg(\alpha_u) + deg(M_u) \leq d$. This allows us to transform the variable set U into a smaller variable set, considering internal linear relations due to the fact that $deg(\alpha_u) \leq d - deg(M_u)$. We stress again that the variable transformation technique can be applied to block ciphers with arbitrary key schedules by considering each round key as independent.

We choose an integral *splitting index* $1 \leq sp \leq d + 1$, and write $U = U' \bigcup U''$, where $U' = \bigcup_{i=1}^{sp-1} U_i$ and $U'' = \bigcup_{i=sp}^{d} U_i$. The observation above implies that the algebraic degree of the variables in U'' (in terms of the key) is bounded by $d - sp$, namely, $deg(\alpha_u) \leq d - sp$, for each $\alpha_u \in U''$. Therefore, we can interpolate each variable of U'' in terms of the key, and express it as

$\alpha_u = \sum_{\{v=(v_1,\ldots,v_\kappa)|wt(v)\leq d-sp\}} \beta_u M_v$, where $\beta_v \in \{0,1\}$ is the coefficient of the monomial $M_v = \prod_{i=1}^{\kappa} x_i^{v_i}$. Note that the coefficients β_v are independent of the key and can be computed during preprocessing. This interpolation transforms the set of variables U'' into the set of variables V, which are low degree monomials in the key bits $V = \{M_v = \prod_{i=1}^{\kappa} x_i^{v_i} | v = (v_1, \ldots, v_\kappa) \wedge wt(v) \leq d - sp\}$. Similarly to the partition of U, we partition the variables of V into subsets according to the degree of their monomials in the key, namely $V_i = \{M_v \in V | deg(M_v) = i\}$. In addition, we define $V_{\leq i} = \bigcup_{j=1}^{i} V_i$. Note that $\alpha_u \in U_i$ is a linear combination of variables in $V_{\leq(d-i)}$.

Recall that our initial set of variables is expressed as $U = U' \bigcup U''$, where $U' = \bigcup_{i=1}^{sp-1} U_i$ and $U'' = \bigcup_{i=sp}^{d} U_i$. This set of variables is transformed via interpolation into a new set of variables $W = U' \bigcup V$.

We compute bounds on sizes of the variables sets as follows:

$$|U'| \leq \binom{256}{\leq sp-1}, \quad |V| \leq \binom{\kappa}{\leq d-sp},$$

$$|W| = |U'| + |V| \leq \binom{256}{\leq sp-1} + \binom{\kappa}{\leq d-sp}.$$

The Variable Transformation Algorithm We now describe the algorithm which interpolates a variable $\alpha_u \in U_i$ in terms of the variable set $V_{\leq(d-i)}$. For the sake of efficiency, the algorithm is performed in two phases, where in the first phase, we evaluate the polynomial α_u in terms of the key for all relevant keys of low Hamming weight and store the results. Note that each evaluation of α_u requires summing on 2^i evaluations of the target bit b. In the second phase, we use the evaluations to interpolate α_u in terms of $V_{\leq(d-i)}$.

1. Allocate a bit array $\boldsymbol{a_1}$ of size $|V_{\leq(d-i)}|$ for the evaluations of α_u.
2. Evaluate α_u for each key with Hamming weight at most $d-i$. Namely, for each key in the set $\{K|wt(K) \leq d-i\}$:
   (a) Evaluate $F(K,C)$ (the target bit) on the subset of 2^i inputs (with the fixed key K) $\{K, C|\bar{u} \wedge C = 0\}$, sum the result over $GF(2)$, and store it in $\boldsymbol{a_1}$.
3. Allocate a bit array $\boldsymbol{a_2}$ of size $|V_{\leq(d-i)}|$ for interpolation of α_u in terms of $V_{\leq(d-i)}$.
4. For each $M_v \in V_{\leq(d-i)}$ (with index ℓ), the coefficient β_v of M_v in α_u is calculated as follows:
   (a) Sum the $2^{wt(v)}$ values of $\boldsymbol{a_1}$ calculated for the subset of keys $\{K|\bar{v} \wedge K = 0\}$, and store the result in $\boldsymbol{a_2}[\ell]$.

The total number of evaluations of b in Step 2 is $2^i \cdot |V_{\leq(d-i)}|$, each requiring $r_1 \cdot 2^{16}$ bit operations. Therefore, the total complexity of this step is $r_1 \cdot 2^{16+i} \cdot |V_{\leq(d-i)}|$. Step 4 requires less than $|V_{\leq(d-i)}| \cdot 2^{d-i}$ bit operations. In total, the interpolation of $\alpha_u \in U_i$ requires $|V_{\leq(d-i)}| \cdot (r_1 \cdot 2^{16+i} + 2^{d-i})$ bit operations.

Since $U'' = \bigcup_{i=sp}^{d} U_i$, we can write the complexity of interpolating all the variables as $\sum_{i=sp}^{d} |U_i| \cdot |V_{\leq(d-i)}| \cdot (r_1 \cdot 2^{16+i} + 2^{d-i})$. A simple way to bound this complexity is

$$|U''| \cdot |V| \cdot (r_1 \cdot 2^{16+d} + 2^{d-sp}) \approx |U''| \cdot |V| \cdot r_1 \cdot 2^{16+d}.$$

In some cases, we can obtain a refined bound by writing the complexity as

$$|U_{sp}| \cdot |V_{\leq(d-sp)}| \cdot (r_1 \cdot 2^{16+sp} + 2^{d-sp}) + \sum_{i=sp+1}^{d} |U_i| \cdot |V_{\leq(d-i)}| \cdot (r_1 \cdot 2^{16+i} + 2^{d-i}) \leq$$

$$|U_{sp}| \cdot |V_{\leq(d-sp)}| \cdot (r_1 \cdot 2^{16+sp} + 2^{d-sp}) + |U''| \cdot |V_{\leq(d-sp-1)}| \cdot (r_1 \cdot 2^{16+d} + 2^{d-sp+1}) \approx$$

$$|U_{sp}| \cdot |V| \cdot (r_1 \cdot 2^{16+sp} + 2^{d-sp}) + |U''| \cdot |V_{\leq(d-sp-1)}| \cdot r_1 \cdot 2^{16+d}.$$

Note that the bound is potentially better than the trivial one of $|U''| \cdot |V| \cdot r_1 \cdot 2^{16+d}$ as $|U_{sp}| \leq \binom{256}{sp}$, which may be smaller than $|U''|$. Moreover $|V_{\leq(d-sp-1)}| \leq \binom{\kappa}{\leq d-sp-1}$, which is smaller than $|V|$.

Transformation of Equations After computing the transformation of variables from U'' to V, we need to apply the actual transformation to every equation over U that we calculated. Namely, we are interested in transforming an equation over the variable set $U = U' \bigcup U''$, into an equation over variable set $W = U' \bigcup V$. Obviously, the coefficients of the variables of U' remain the same, and we need to apply the transformation for every variable $\alpha_u \in U''$. The complexity of transforming a single variable $\alpha_u \in U_i$ in a single equation is simply equal to its number of coefficients over V, namely $|V_{\leq(d-i)}|$. Therefore, the complexity of transforming all the variables $\alpha_u \in U''$ in an equation is $\sum_{i=sp}^{d} |U_i| \cdot |V_{\leq(d-i)}|$. A simple upper bound on this complexity is

$$|U''| \cdot |V|.$$

Similarly to the variable transformation algorithm, a refined upper bound can be calculated as

$$|U_{sp}| \cdot |V| + |U''| \cdot |V_{\leq(d-sp-1)}|.$$

In total, if we transform e equations, the complexity calculations above are multiplied by e.

Finally, we observe that the splitting index determines the complexity of the variable and equation transformation algorithms. Furthermore, the splitting index also determines $|W|$, which in turn determines the number of equations e. In general, we will choose sp in order to minimize $|W|$, which in turn minimizes the data and time complexity of the attack.

5.2 Details of the Optimized Interpolation Attack

Given an instance of LowMC with a 256-bit block, a key size of κ, and m Sboxes per layer, we interpolate a target bit b through the final r_1 rounds of the cipher. Let U, U', U'', V and W be as defined above, and let $e \approx |W|$ denote the number of equations. Assume S is a sufficiently large subspace of plaintexts, such that it contains e smaller subspaces $S_1, \ldots, S_e$ whose high-order differential on b is a constant value (independent of the key).

The preprocessing phase of the optimized attack in described below.

Preprocessing:

1. Compute an e-bit array of free coefficients for $e \approx |U'|$ equations, denoted by $\boldsymbol{a_0}$: evaluate b on the subset of inputs (plaintexts) of S (with the key set to zero), and obtain a bit array of size $|S|$. Then, calculate the free coefficients by applying the Moebius transform to the bit array, and copy the values of sums over $S_1, \ldots, S_e$ to $\boldsymbol{a_0}$.
2. Calculate the $|U|$ vectors $\{u|\alpha_u \in U\}$: This is done by first calculating the $256+3m$ monomials past the first Sbox layer, and multiplying them in unordered tuples (with no repetition) of size up to 2^{r_1-1}(as described in Section 5.1).

Step 1 involves $|S|$ evaluations of the encryption scheme and one application of the Moebius transform on a vector of size S. Altogether, it requires $|S| \cdot 2^{19} + \log(|S|) \cdot |S| \approx |S| \cdot 2^{19}$ bit operations (as $\log(|S|) \ll 2^{19}$). Step 2 requires $|U|$ monomial multiplications, each monomial can be represented with a 256-bit array, and therefore this step requires $2^8 \cdot |U|$ bit operations.

A summary of the complexity analysis of the preprocessing phase is as follows.
Step 1: $2^{19} \cdot |S|$
Step 2: $2^8 \cdot |U|$

In terms of memory, Step 1 requires $|S|$ bits, while Step 2 requires $2^8 \cdot |U|$ bits.

Online:

1. Ask for the encryptions of the plaintexts in S and store the ciphertexts in a table.

2. Allocate a bit vector of size $|S|$ for the storage of the vectors $\boldsymbol{a_\ell}$ (the ℓ'th column of the matrix A in the basic attack).
3. Allocate an $e \times |W|$ matrix E over $GF(2)$, representing the (reduced) equation system on W. The matrix is vertically decomposed into two smaller matrices: E_1 of size $e \times |U'|$ and E_2 of size $e \times |V|$.
4. For each $\{M_u | \alpha_u \in U\}$ with an index ℓ:
   (a) For each ciphertext C_t, calculate $\boldsymbol{a_\ell[t]}$ by evaluating $M_u(C_t)$.
   (b) Use the Moebius transform to sum over all subspaces of $\boldsymbol{a_\ell}$.
   (c) If $\alpha_u \in U'$, populate column ℓ of E_1: For each subspace S_j in S, namely $S_1, \ldots, S_e$, obtain its corresponding sum from $\boldsymbol{a_\ell}$ and copy it to $E_1[j][\ell]$.
   (d) Otherwise, $\alpha_u \in U''$:
      i. Given that $\alpha_u \in U_i$, interpolate the coefficients of $V_{\leq(d-i)}$ in α_u as described in Section 5.1.
      ii. For each subspace S_j in S, obtain its corresponding boolean sum from $\boldsymbol{a_\ell}$ (the coefficient of α_u over U). If the sum is 1, then add (over $GF(2)$) the interpolated coefficients into their indexes in $E_2[j]$ (as described in Section 5.1).
5. Solve the equation system $E\boldsymbol{x} = \boldsymbol{a_0}$, where $\boldsymbol{x}$ represents the vector of variables of $W = U' \bigcup V$ and $\boldsymbol{a_0}$ is the vector of free coefficients calculated in preprocessing Step 1.
6. Deduce the κ-bit secret key, which is simply given by the monomials V_1 (namely, the monomials of degree 1 in V).

The complexity of Step 1 is $|S|$ encryptions, or $|S| \cdot 2^{19}$ bit operations. In Step 4, we iterate over $|U|$ monomials, where for each one we first evaluate $M_u(C_t)$ for each ciphertext in Step 4.a. Each such evaluation can be performed with d bit operations (as $deg(M_u) \leq d$), and thus monomial evaluations require about $d \cdot |S| \cdot |U|$ bit operations. Next, we apply the Moebius transform in Step 4.b, requiring about $\log(|S|) \cdot |S|$ bit operation, and therefore the complexity of all the transforms is about $\log(|S|) \cdot |S| \cdot |U|$. The complexity of interpolating all the variables in Step 4.d.i, is bounded in Section 5.1 by $|U''| \cdot |V| \cdot r_1 \cdot 2^{16+d}$. The complexity of Step 4.d.ii (over all $\alpha_u \in U''$) is bounded in Section 5.1 by $e \cdot |U''| \cdot |V| \approx |W| \cdot |U''| \cdot |V|$.

The complexity of Step 5 is $|W|^3$ bit operations using Gaussian elimination. A summary of the complexity analysis of the online phase is as follows. Since we generally do not have a good bound for $|U''|$, we simply replace it with $|U|$ (as $|U''| \leq |U|$), and further assume that $e \approx |W|$.

Step 1: $|S| \cdot 2^{19}$
Step 2: $|S|$
Step 3: $|W| \cdot |W|$
Step 4.a: $d \cdot |S| \cdot |U|$
Step 4.b: $\log(|S|) \cdot |S| \cdot |U|$
Step 4.c: $|U'| \cdot |W|$
Step 4.d.i: $|U| \cdot |V| \cdot r_1 \cdot 2^{16+d}$
Step 4.d.ii: $|W| \cdot |U| \cdot |V|$

Step 5: $|W|^3$
Step 6: negligible

Alternatively, we can use the refined complexity bounds for steps 4.d.i and 4.d.ii, as calculated in Section 5.1.
Step 4.d.i: $|U_{sp}| \cdot |V| \cdot (r_1 \cdot 2^{16+sp} + 2^{d-sp}) + |U| \cdot |V_{\leq(d-sp-1)}| \cdot r_1 \cdot 2^{16+d}$
Step 4.d.ii: $|W| \cdot (|U_{sp}| \cdot |V| + |U| \cdot |V_{\leq(d-sp-1)}|)$

The total data complexity of the algorithm is $|S|$ chosen plaintexts. The total time complexity is dominated by steps 4 and 5, as calculated above. The memory complexity is potentially dominated by a few steps: the storage of variables in preprocessing that requires $2^8 \cdot |U|$ bits, the storage of ciphertexts in Step 1 that requires $2^8 \cdot |S|$ bits, and the storage of E in Step 3 that requires $|W| \cdot |W|$ bits.

6 Optimized Interpolation Attacks on LowMC-80

In this section we apply the optimized interpolation attack on LowMC-80, for which $\kappa = 80$ and $m = 49$.

6.1 A 9-Round Attack

As in the basic attack described in Section 4.4, we select the target bit b in $Z_6|IP$, using subspaces of dimension 32 to obtain the equations. We interpolate through $r_1 = 2$ rounds, implying that $d = 2^{r_1} = 4$. Therefore $|U| = \binom{256+3m}{\leq 2^{r_1}-1} = \binom{403}{\leq 2} \approx 2^{16.5}$.

As described at the beginning of Section 5, we use $sp = 2$. We compute the size of the relevant variable sets $|U'| \leq \binom{256}{\leq sp-1} = \binom{256}{\leq 1} \approx 2^8$, $|V| \leq \binom{\kappa}{\leq d-sp} = \binom{80}{\leq 2} < 2^{12}$, $|W| = |U'| + |V| < 2^{12}$.

We choose a subspace S of dimension 35 from $X_0|IP$, containing $\binom{35}{32} > 2^{12} > |W|$ 32-dimensional subspaces, which should suffice for the attack.

In terms of time complexity, the analysis of the critical steps of the attack is as follows:
Step 4.a: $d \cdot |S| \cdot |U| \approx 4 \cdot 2^{35} \cdot 2^{16.5} = 2^{53.5}$
Step 4.b: $\log(|S|) \cdot |S| \cdot |U| \approx 35 \cdot 2^{35} \cdot 2^{16.5} = 2^{56.5}$
Step 4.c: $|U'| \cdot |W| \approx 2^8 \cdot 2^{12} = 2^{20}$
Step 4.d.i: $|U| \cdot |V| \cdot r_1 \cdot 2^{16+d} \approx 2^{16.5} \cdot 2^{12} \cdot 2 \cdot 2^{20} = 2^{49.5}$
Step 4.d.ii: $|W| \cdot |U| \cdot |V| \approx 2^{12} \cdot 2^{16.5} \cdot 2^{12} = 2^{40.5}$
Step 5: $|W|^3 \approx 2^{12 \cdot 3} = 2^{36}$

In total, the time complexity of the optimized 9-round attack is about 2^{57} bit operations (or $2^{57-19} = 2^{38}$ encryptions), mostly dominated by Step 4.b. The data complexity is 2^{35} chosen plaintexts. The memory complexity is dominated by the storage of ciphertexts in Step 1, and is about $|S| \cdot 2^8 = 2^{43}$ bits.

We note that while the improvement of the optimized attack compared to the basic one is rather moderate for the 9-round attack, the effect of our optimizations is more pronounced in the attacks described next, as the reduction in the number of variables becomes more significant (a comparison for the attack on full LowMC-128 is at the end of Section 7.2).

6.2 A 10-Round Attack

Similarly to the 9-round attack, in order to attack 10 rounds of LowMC-80, we select the target bit b in $Z_6|IP$, using subspaces of dimension 32 to obtain the equations. We interpolate through $r_1 = 3$ rounds, implying that $d = 2^{r_1} = 8$. Therefore $|U| = \binom{256+3m}{\leq 2^{r_1}-1} = \binom{403}{\leq 4} < 2^{30.5}$.

In this attack we use $sp = 4$, and compute the size of the relevant variable sets $|U'| \leq \binom{256}{\leq sp-1} = \binom{256}{\leq 3} \approx 2^{21.5}$, $|V| \leq \binom{\kappa}{\leq d-sp} = \binom{80}{\leq 4} < 2^{21}$, $|W| = |U'| + |V| < 2^{22.5}$. We use the refined analysis for steps 4.d.i and 4.d.ii, and thus we also calculate $|U_{sp}| = |U_4| = \binom{256}{4} < 2^{27.5}$ and $|V_{\leq(d-sp-1)}| = \binom{80}{\leq 3} < 2^{16.5}$.

We choose a subspace S of dimension 39 from $X_0|IP$, containing $\binom{39}{32} > 2^{23} > |W|$ 32-dimensional subspaces.

In terms of time complexity, the analysis of the critical steps of the attack is as follows (using the refined analysis for steps 4.d.i and 4.d.ii):
Step 4.a: $d \cdot |S| \cdot |U| \approx 8 \cdot 2^{39} \cdot 2^{30.5} = 2^{72.5}$
Step 4.b: $\log(|S|) \cdot |S| \cdot |U| \approx 39 \cdot 2^{39} \cdot 2^{30.5} \approx 2^{75}$
Step 4.c: $|U'| \cdot |W| \approx 2^{21.5} \cdot 2^{22.5} = 2^{44}$
Step 4.d.i: $|U_{sp}| \cdot |V| \cdot (r_1 \cdot 2^{16+sp} + 2^{d-sp}) + |U| \cdot |V_{\leq(d-sp-1)}| \cdot r_1 \cdot 2^{16+d} \approx 2^{27.5} \cdot 2^{21} \cdot (3 \cdot 2^{20} + 2^4) + 2^{30.5} \cdot 2^{16.5} \cdot 3 \cdot 2^{24} \approx 2^{70} + 2^{72.5} \approx 2^{73}$
Step 4.d.ii: $|W| \cdot (|U_{sp}| \cdot |V| + |U| \cdot |V_{\leq(d-sp-1)}|) \approx 2^{22.5} \cdot (2^{27.5} \cdot 2^{21} + 2^{30.5} \cdot 2^{16.5}) \approx 2^{22.5} \cdot (2^{48.5} + 2^{47}) \approx 2^{71.5}$
Step 5: $|W|^3 \approx 2^{22.5 \cdot 3} = 2^{67.5}$

In total, the time complexity of the optimized 10-round attack is about 2^{76} bit operations (or 2^{57} encryptions), mostly dominated by Step 4.b. The data complexity is 2^{39} chosen plaintexts. The memory complexity is dominated by the storage of ciphertexts in Step 1, and is about $2^8 \cdot |S| = 2^{47}$ bits (note that the storage of E requires $2^{22.5 \cdot 2} = 2^{45}$ bits).

6.3 An Attack on Full LowMC-80 for Weak Instances

The 9 and 10-round attacks described above can be extended by an additional round with negligible cost for a subset of weak instances containing a fraction of about 2^{-38} of all instances. In particular, this implies that about 2^{-38} of the instances of full 11-round LowMC-80 can be attacked significantly faster than exhaustive search.

Consider the 10-round attack: as shown above, we can construct an efficient high-order differential property for any choice of target bit of $Z_6|IP$, and also for any linear combination of the bits of $Z_6|IP$. When considering interpolation

from the decryption side on a full 11-round instance, we can efficiently interpolate the polynomial $F_K(C)$ for any bit of $Z_7|IP$, or any linear combination of the bits of $Z_7|IP$. Assume that there exists a linear dependency between the 109 bits of $Z_6|IP$ and the 109 bits of $Z_7|IP$. In this case, the linear combination in terms of $Z_6|IP$ does not go through an Sbox in round 8. Therefore, it is possible to extend the high-order differential property on this linear combination by another round with essentially no extra cost, and choose the target bit for interpolation to be the corresponding linear combination on the bits of $Z_7|IP$. The existence of this linear dependency is determined by the affine layer of round 7 (the transformation between Z_6 and X_7), and assuming that random invertible matrices behave roughly the same (with respect to the event considered) as random matrices, the probability of this event is about $2^{109+109-256} = 2^{-38}$ (over the choice of the 7'th affine layer).

We note that there exists an additional subset of weak instances of about the same size since the described attacks can also be mounted using chosen ciphertexts (where interpolation is performed on the decrypted plaintexts). In this case, the weakness of a given instance is determined by the choice of the third affine layer.

7 Optimized Interpolation Attacks on LowMC-128

In this section we apply the optimized interpolation attack on LowMC-128, for which $\kappa = 128$ and $m = 63$.

7.1 An 11-Round Attack and Weak Instances of LowMC-128

We describe our attack on 11-round LowMC-128 and then extend it to full LowMC-128 for weak instances. We select the target bit b in $Z_7|IP$, and interpolate through $r_1 = 3$ rounds, implying that $d = 2^{r_1} = 8$. Therefore $|U| = \binom{256+3m}{\leq 2^{r_1}-1} = \binom{445}{\leq 4} < 2^{31}$.

In this attack we use $sp = 4$, and compute the size of the relevant variable sets $|U'| \leq \binom{256}{\leq sp-1} = \binom{256}{\leq 3} \approx 2^{21.5}$, $|V| \leq \binom{\kappa}{\leq d-sp} = \binom{128}{\leq 4} \approx 2^{23.5}$, $|W| = |U'| + |V| \approx 2^{24}$.

For the high-order differential property, we use subspaces of dimension $2^6 = 64$ whose bits are not multiplied together in the first round. The outcome of such a high-order differential is a constant (independent of the key) for $1+6 = 7$ rounds, and this property can be extended beyond the 8'th Sbox layer when selecting the target bit from $Z_7|IP$.

Since $|W| \approx 2^{24}$, we require roughly the same number of 64-dimensional subspaces to construct the equation system and mount the attack. Therefore, we take a larger subspace of dimension 70, containing $\binom{70}{64} > 2^{24} \approx |W|$ 64-dimensional subspaces. As $X_0|IP$ contains only 67 bits, we choose the subspace from these 67 bits and additional 3 bits in $X_0|SP$, contained in 1 active Sbox. Since the active Sbox is non-linear, we guess the 3 linear key expressions that are

added to its input, which allow us to construct the required $\approx 2^{24}$ 64-dimensional subspaces from a 70-dimensional subspace after the first Sbox layer.

The guess of the 3 key bits can be avoided by selecting the $70 - 64 = 6$ constant bits of the 64-dimensional subspaces from the 67 bits of $X_0|IP$ in the 70-dimensional subspace. This restriction keeps the selected Sbox fully active in all subspaces, and thus the linear subspace after the first Sbox layer (at Z_0) is independent of the key bits. The number of such restricted 64-dimensional subspaces is $\binom{67}{6} > 2^{24} \approx |W|$, and hence they should suffice for the attack.

Finally, we notice that the Moebius transforms (Step 4.b) can be optimized due to the way that we chose the subspaces in S, as for all of them, 3 specific bits of $X_0|SP$ are active. In order to exploit this, we perform the Moebius transform on a 2^{70} bit vector in two phases: in the first phase, we partition the 2^{70} big subspace into 2^{67} 3-dimensional subspaces according to the 67 bits of $X_0|IP$, and sum on all of them in time 2^{70}, obtaining a vector of size 2^{67}. In the second phase, we perform the Moebius transform on the 2^{67} vectors computed in the first phase. Therefore, the complexity of a single Moebius transform is reduced from $70 \cdot 2^{70} \approx 2^{76}$ to $2^{70} + 67 \cdot 2^{67} \approx 2^{73}$. The complexity of online Step 4.b now becomes $|U| \cdot 2^{73} \approx 2^{104}$ bit operations.

The time complexity analysis of the critical steps of the attack is as follows:
Step 4.a: $d \cdot |S| \cdot |U| \approx 8 \cdot 2^{70} \cdot 2^{31} = 2^{104}$
Step 4.b: 2^{104} (as noted above)
Step 4.c: $|U'| \cdot |W| \approx 2^{21.5} \cdot 2^{24} = 2^{45.5}$
Step 4.d.i: $|U| \cdot |V| \cdot r_1 \cdot 2^{16+d} \approx 2^{31} \cdot 2^{23.5} \cdot 3 \cdot 2^{24} \approx 2^{80.5}$
Step 4.d.ii: $|W| \cdot |U| \cdot |V| \approx 2^{24} \cdot 2^{31} \cdot 2^{23.5} = 2^{78.5}$
Step 5: $|W|^3 \approx 2^{24 \cdot 3} = 2^{72}$

In total, the time complexity of the attack is about 2^{105} bit operations, dominated by steps 4.a and 4.b. The data complexity is 2^{70} chosen plaintexts. The memory complexity is dominated by the storage of ciphertexts in Step 1, and is about $|S| \cdot 2^8 = 2^{78}$ bits.

Extending the Attack to Full LowMC-128 for Weak Instances Similarly to the attacks on LowMC-80, the 11-round attack on LowMC-128 can be extended by an additional round with no increase in complexity for a subset of weak instances. However, the fraction of these instances is much smaller, as the I-part of LowMC-128 contains only 67 bits, and is smaller than the one of LowMC-80. A similar analysis to the one of Section 6.3 shows that the fraction of such weak instances for LowMC-128 is roughly $2^{67+67-256} = 2^{-122}$. As noted in the Introduction, this attack does not violate the formal security claims of the LowMC designers.

7.2 An Attack on Full LowMC-128

We now describe our attack on full (12-round) LowMC-128. This attack is more marginal than the previous attacks, and we have to use essentially all of our

previously described optimizations, as well as new ones in order to obtain an attack which is faster than exhaustive search.

In order to attack 12 rounds of LowMC-128, we extend the interpolation of the 11-round attack past another round, interpolating $Z_7|IP$ through $r_1 = 4$ Sbox layers, and hence $d = 2^4 = 16$, $|U| = \binom{256+3m}{\leq 2^{r_1}-1} = \binom{445}{\leq 8} \approx 2^{55}$.

In this attack we use $sp = 8$, and compute the size of the relevant variable sets $|U'| \leq \binom{256}{\leq sp-1} = \binom{256}{\leq 7} \approx 2^{43.5}$, $|V| \leq \binom{\kappa}{\leq d-sp} = \binom{128}{\leq 8} \approx 2^{40.5}$, $|W| = |U'| + |V| \approx 2^{44}$. We use the refined analysis for steps 4.d.i and 4.d.ii, and thus we also calculate $|U_{sp}| = |U_8| = \binom{256}{8} < 2^{48.5}$ and $|V_{\leq (d-sp-1)}| = \binom{128}{\leq 7} < 2^{36.5}$.

The High-Order Differential Property We can try to mount the attack with high-order differentials on subspaces of dimension 64 for the target bit in $Z_7|IP$, but this results in an attack which is at best very marginally faster than exhaustive search. The main new optimization introduced in this attack is the use of reduced subspaces of dimension 60. Obviously, the result of a high-order differentiation over such a subspace is not a constant, but (as we show next) its algebraic degree in the key bits is bounded by 8. Consequently, the resultant function (polynomial) of each high-order differentiation can be expressed in terms of our reduced variable set $V = |V_{\leq (8)}|$. This polynomial can be interpolated during preprocessing and does not contribute additional variables to the equation system.

We select a big subspace S of dimension 73 that contains all the 67 bits of $X_0|IP$ and 6 additional bits of 2 active Sboxes in $X_0|SP$, and (similarly to the 11-round attack) define the 60-dimensional subspaces according to their $73-60 = 13$ constant bits in $X_0|IP$. The number of such subspaces is $\binom{67}{13} > 2^{44} \approx |W|$, and therefore they should suffice for the attack.

In order to show that the result of a high-order differentiation of the target bit in $Z_7|IP$ over a selected 60-dimensional is of degree 8 in the key bits, consider the state Z_0 obtained after the first Sbox layer. The algebraic degree of the target bit b (selected from $Z_7|IP$) in Z_0 is bounded by $2^6 = 64$. As the linear subspace undergoes a one-to-one transformation in the first Sbox layer (through the fully active 2 Sboxes), it remains a linear subspace in Z_0. Therefore, the algebraic degree of the high-order differentiation in the bits of Z_0 and the key is upper-bounded by $64-60 = 4$. Since each bit of Z_0 is a polynomial in the key of degree (at most) 2, the algebraic degree of the high-order differentiation in the bits of the key is upper-bounded by $4 \cdot 2 = 8$, as claimed.

The Preprocessing Phase The main change in this attack compared to the one of Section 5.2 is in preprocessing Step 1, where in addition to interpolating the $e \approx |W|$ free coefficients, we interpolate the $e \cdot |V| \approx |W| \cdot |V|$ coefficients of V (since we selected 60-dimensional subspaces instead of 64-dimensional subspaces). The modified preprocessing step is described below. It is similar to the variable transformation algorithm of Section 5.1, interpolating first over the plaintexts and then over the keys. Note that the matrix E of linear equations is allocated and initialized already at this stage.

1. Allocate an $e \times |W|$ matrix E over $GF(2)$, representing the (reduced) equation system on W. The matrix is vertically decomposed into two smaller matrices: E_1 of size $e \times |U'|$ and E_2 of size $e \times |V|$.
2. Allocated an $e \cdot |V|$ evaluation matrix EV.
3. Allocate a $|S| = 2^{73}$ bit array $\boldsymbol{a_1}$ for the evaluations of the target bit b.
4. For each key in the set $\{K|wt(K) \leq 8\}$ (with index ℓ):
   (a) Evaluate b (the target bit) on the set S of 2^{73} inputs (with the fixed key K) and store the result in $\boldsymbol{a_1}$.
   (b) Apply the Moebius transform on $\boldsymbol{a_1}$.
   (c) Populate column ℓ of EV: For each subspace S_j in S, namely $S_1, \ldots, S_e$, obtain its corresponding sum from $\boldsymbol{a_1}$ and copy it to $E_1[j][\ell]$.
5. For each equation $1, \ldots, e$ (with index j):
   (a) For each $M_v \in V_{\leq 8} = V$ (with index ℓ):
      i. Sum the $2^{wt(\bar{v})}$ values of $EV[j]$ calculated for the subset of keys $\{K|\bar{v} \wedge K = 0\}$, and store the result in $E_2[j][\ell]$.

We first note that similarly to the 11-round attack, the complexity of the Moebius transform can be optimized (due to the way that we selected the subspaces) in a 2-step process from $73 \cdot 2^{73}$ to $2^{73} + 67 \cdot 2^{67} \approx 2^{74}$.

We analyze the complexity of the computationally heavy steps 4 and 5. The complexity of Step 4.a (for all $\{K|wt(K) \leq 8\}$) is $|V| \cdot |S| \cdot 2^{19} \approx 2^{40.5} \cdot 2^{73} \cdot 2^{19} = 2^{132.5}$. The complexity of Step 4.b (using the optimized Moebius transform) is $|V| \cdot 2^{74} \approx 2^{114.5}$. The complexity of Step 4.c is $e \cdot |V| \approx |W| \cdot |V| \approx 2^{44} \cdot 2^{40.5} = 2^{84.5}$. The complexity of Step 5.a.i is bounded by $e \cdot |V| \cdot 2^8 \approx 2^{44} \cdot 2^{40.5} \cdot 2^8 = 2^{92.5}$. In total, Step 4.a dominates the time complexity, which is about $2^{132.5}$ bit operations.

Analysis of the Full Attack In terms of time complexity, the analysis of the critical steps of the online attack is as follows (using the optimized Moebius transform and the refined analysis for steps 4.d.i and 4.d.ii):
Step 4.a: $d \cdot |S| \cdot |U| \approx 16 \cdot 2^{73} \cdot 2^{55} = 2^{132}$
Step 4.b: $|U| \cdot 2^{74} \approx 2^{129}$
Step 4.c: $|U'| \cdot |W| \approx 2^{43.5} \cdot 2^{44} = 2^{87.5}$
Step 4.d.i: $|U_{sp}| \cdot |V| \cdot (r_1 \cdot 2^{16+sp} + 2^{d-sp}) + |U| \cdot |V_{\leq (d-sp-1)}| \cdot r_1 \cdot 2^{16+d} \approx 2^{48.5} \cdot 2^{40.5} \cdot (4 \cdot 2^{24} + 2^8) + 2^{55} \cdot 2^{36.5} \cdot 4 \cdot 2^{32} \approx 2^{115} + 2^{125.5} \approx 2^{125.5}$
Step 4.d.ii: $|W| \cdot (|U_{sp}| \cdot |V| + |U| \cdot |V_{\leq (d-sp-1)}|) \approx 2^{44} \cdot (2^{48.5} \cdot 2^{40.5} + 2^{55} \cdot 2^{36.5}) \approx 2^{44} \cdot (2^{89} + 2^{91.5}) \approx 2^{136}$
Step 5: $|W|^3 \approx 2^{44 \cdot 3} = 2^{132}$

The online phase complexity is about 2^{136} dominated by[3] Step 4.d.ii. The total complexity of the attack is less than 2^{137} bit operations, which is about

[3] We note that the analysis of Step 4.d.ii can be refined further, and its actual complexity is lower by a factor between 2 and 4. Moreover, the actual algorithm of this step can be optimized, but we do not consider such low-level optimizations here for the sake of simplicity.

$2^{128+19-137} = 2^{10}$ times faster than exhaustive search (including the preprocessing phase, whose complexity is about $2^{132.5}$). The data complexity of the attack is 2^{73} chosen plaintexts. The memory complexity is dominated by the storage of E, whose size is about $|W| \cdot |W| \approx 2^{88}$ bits.

Note that without the variable transformation, merely Step 5 (Gaussian elimination) would require about $2^{55 \cdot 3} = 2^{165}$ bit operations, which is much slower than exhaustive search.[4]

8 Conclusions

In this paper, we introduced new techniques for interpolation attacks, including a new variable transformation algorithm that can lead to savings in their data and time complexities. We applied the optimized interpolation attack to LowMC, and refuted the claims of the designers regarding the security level of both the 80 and 128-bit key variants. As a future work item, it will be interesting to optimize our techniques further and apply them to additional block ciphers.

References

1. M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner. Ciphers for MPC and FHE. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 430–454. Springer, 2015.
2. I. Dinur and A. Shamir. Cube Attacks on Tweakable Black Box Polynomials. In A. Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 278–299. Springer, 2009.
3. M. Hell, T. Johansson, A. Maximov, and W. Meier. The Grain Family of Stream Ciphers. In M. J. B. Robshaw and O. Billet, editors, *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 179–190. Springer, 2008.
4. T. Jakobsen and L. R. Knudsen. The Interpolation Attack on Block Ciphers. In E. Biham, editor, *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings*, volume 1267 of *Lecture Notes in Computer Science*, pages 28–40. Springer, 1997.
5. A. Joux. *Algorithmic Cryptanalysis*. Chapman & Hall/CRC, 1st edition, 2009. Pages 285-286.

[4] Solving the equation system remains slower than exhaustive search even when using more advanced algorithms which are based on Strassen's algorithm [9], requiring about $2^{55 \cdot 2.8} = 2^{154}$ bit operations. While there are known algorithms that perform better in theory, most of them are very complex and inefficient in practice.

6. L. R. Knudsen. Truncated and Higher Order Differentials. In B. Preneel, editor, *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*, volume 1008 of *Lecture Notes in Computer Science*, pages 196–211. Springer, 1994.
7. X. Lai. Higher Order Derivatives and Differential Cryptanalysis. In *"Symposium on Communication, Coding and Cryptography", in honor of James L. Massey on the occasion of his 60'th birthday*, pages 227–233, 1994.
8. T. Shimoyama, S. Moriai, and T. Kaneko. Improving the Higher Order Differential Attack and Cryptanalysis of the *KN* Cipher. In E. Okamoto, G. I. Davida, and M. Mambo, editors, *Information Security, First International Workshop, ISW '97, Tatsunokuchi, Japan, September 17-19, 1997, Proceedings*, volume 1396 of *Lecture Notes in Computer Science*, pages 32–42. Springer, 1997.
9. V. Strassen. Gaussian Elimination is not Optimal. *Numerische Mathematik*, 13:354–356, 1969.

Appendix A

For a binary vector $x = (x_1, \ldots, x_m) \in \mathbb{F}_2^m$, the *weight* of x is the Hamming weight $wt(x) = \sum_{i=1}^{m} x_i$.

A *Boolean function of m variables* is a function $f\colon \mathbb{F}_2^m \to \mathbb{F}_2$, and we denote by $\mathcal{B}_m$ the set of all Boolean functions of m variables. Among the classical representations of Boolean functions, the one which is most usually used in cryptography and coding is the m-variable polynomial representation over $\mathbb{F}_2$ of the form

$$f(x_1, \ldots, x_m) = \sum_{u \in \mathbb{F}_2^m} \lambda_u \left(\prod_{i=1}^{m} x_i^{u_i} \right) , \ \lambda_u \in \mathbb{F}_2 \ .$$

It is called the *Algebraic Normal Form* (ANF) of f.

The *algebraic degree* of f, denoted as $\deg f$, is the maximum value of $wt(u)$ such that $\lambda_u \neq 0$.

Definition 4 ([51]). *The permutation f is said to be almost bent (AB) if the Walsh transform*

$$\mu_f(\alpha, \beta) = \sum_{x \in \mathbb{F}_2^n} (-1)^{<\alpha, x> \oplus <\beta, f(x)>}$$

is equal to either 0 or $\pm 2^{\frac{n+1}{2}}$ when $\alpha, \beta \in \mathbb{F}_2^n$ and $(\alpha, \beta) \neq (0, 0)$, here $<,>$ is the inner product on $\mathbb{F}_2^n$.

An AB permutation provides optimum resistance against both differential and linear cryptanalysis, however it exists only when n is odd.

Curriculum Vitae

Qingju Wang was born in Shandong Province, China. She received B.Sc and M.Sc in Applied Mathematics from Linyi University and Central South University, China in July 2003 and March 2006 respectively. Then she worked as assistant Professor in Department of Mathematics, Shaoxing University, China. In September 2009, She started her PhD in Shanghai Jiao Tong University. In October 2010, She visited COSIC as an exchange student and joined COSIC as a joint PhD student of KU Leuven and Shanghai Jiao Tong University from February 2012.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING (ESAT)
COMPUTER SECURITY AND INDUSTRIAL CRYPTOGRAPHY (COSIC)
Kasteelpark Arenberg 10, Bus 2452
B-3001 Heverlee
qingju.wang@esat.kuleuven.be
www.esat.kuleuven.be/cosic/