KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

# Techniques for Block Cipher Cryptanalysis

**Yunwen LIU**

Supervisor:
Prof. dr. ir. Vincent Rijmen

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

September 2018

# Techniques for Block Cipher Cryptanalysis

**Yunwen LIU**

Examination committee:
Prof. dr. ir. Joos Vandewalle, chair
Prof. dr. ir. Vincent Rijmen, supervisor
Prof. dr. ir. Bart Preneel
Prof. dr. ir. Luc Van Eycken
Prof. dr. ir. Joan Daemen
(Radboud University)
Prof. dr. Gregor Leander
(Ruhr-Universität Bochum)

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

September 2018

# Preface

不知周之梦为蝴蝶与？蝴蝶之梦为周与？

—《齐物论》*The Butterfly Dream* by Zhuangzi

The doctoral study to me is just like the Boléro: through a spiral circulation it goes on and on, and now it arrives at the last few bars of the melody. I would like to thank everyone who has supported, encouraged and accompanied me, and all the gratefulness shall never fade in my memory.

My supervisor Prof. Vincent Rijmen is the person to whom I own a deep debt of gratitude and thankfulness. The excellence, kindness, patience of Prof. Rijmen is beyond my greatest admiration and respect: becoming his student is definitely one of the best things that ever happened in my life. Being a supervisor who sincerely cares for the development of his students, he keeps enlightening me with wisdom, advices and practical suggestions. Whenever I look back, every step that was firmly taken was always under his guidance.

I would like to thank the jury members of the committee, Prof. Joan Daemen, Prof. Gregor Leander, Prof. Bart Preneel, Prof. Joos Vandewalle, and Prof. Luc Van Eycken, for their time in reading the manuscript, and their valuable suggestions and comments to improve my thesis and research. Especially, I am more than grateful to Prof. Bart Preneel, who together with my supervisor gave me the opportunity to start the doctoral programme, and generously offers advice and help during my stay in COSIC.

In the doctoral study, it is my pleasure and honour to work with many excellent researchers. The valuable experience in our collaborations has positively influenced my research and professional skills, as well as my understanding of an academic career. For that, I would like to thank Tomer Ashur, Hua Chen, Glenn De Witte, Itai Dinur, Limin Fan, Jingyi Feng, Shaojing Fu, Gregor Leander, Chao Li, Chaoyun Li, Wei Li, Guoqiang Liu, Ya Liu, Willi Meier, Longjiang Qu, Adrián Ranea, Yu Sasaki, Ling Song, Bing Sun, Zhi Tao, Gaoli Wang, and

Qingju Wang, without whom this thesis could never be possible.

I enjoy every single frame of Leuven and Belgium in my memory, including the gloomy and naughty weather. To me, Leuven is a hometown which witnesses my growth, and the home address is Kasteelpark Arenberg 10. I would like to thank our dear Péla Noë, and the administration team Elsy Vermoesen, Wim Devroye, Saartje Verheyen and Dana Brouckmans, for their dedications in administrative works to keep us free of worry. Without them, it is easy for me to imagine how messy life could be. Moreover, my heartfelt thanks go to Bing Sun for his guidance and help both in Changsha and in Leuven since 2009; to Tomer Ashur for being a great friend, collaborator and office mate; to Qingju Wang for her kindness in walking me through the early days of work and life in Leuven; to Bart Mennink and Atul Luykx for their advices in research and presentation; and to Sara Cleemput and Yu-long Chen for the Dutch abstract of my thesis. I would also like to thank Hua Chen, Wenying Zhang, Wei Li, Bohan Yang, Ren Zhang, Chaoyun Li, Aysajan Abidin and Kent Chuang; their friendship and hospitality largely eased my homesick in a foreign country. Apart from being an outstanding team at work, COSIC has also organised many interesting activities, where many memorable hours were spent together with Victor Arribas Abril, Günes Acar, Abdelrahaman Aly, Arthur Beckers, Begül Bilgin, Dušan Božilov, Carl Bootland, Ruan De Clercq, Thomas De Cnudde, Lauren De Meyer, Eduard Marín Fàbregas, Miloš Grujić, Ilia Iliashenko, Pieter Maene, Jose M. Bermudo Mera, Mustafa A. Mustafa, Fatemeh Shirazi, Danilo Šijačić, Iraklis Symeonidis, Alan Szepieniec, Kerem Varıcı, Lennert Wouters, and many old and new COSICs.

In my home country, Prof. Chao Li and Prof. Longjiang Qu provided their kindest help and support during my bachelor and master study, also for several research visits in the past four years. Without their encouragement, I could never imagine myself starting to pursue a doctoral degree. Outside COSIC, I have also received a lot of help from Ralph Ankele, Patrick Derbez, Prof. Jian Guo, Stefan Kölbl, Prof. Christian Rechberger, Yu Sasaki, Ernst Schulte-Geers, Tyge Tiessen and Prof. Bin Zhang. It is my great pleasure to talk with them through conferences, workshops and emails, and their advice has greatly improved my research. In addition, I would like to acknowledge the China Scholarship Council (CSC) and COSIC for the financial support.

Finally, I would like to express my sincere gratitude to my parents for their unconditional support, sacrifice and love. They are my source of courage and strength, and raise me to be a happy person.

Yunwen Liu

in Leuven, August 2018

# Abstract

This thesis mainly focuses on cryptanalytic techniques for block ciphers, which benefits the design of cryptographic components as well.

For ARX structures, we propose a novel analytic method called rotational-XOR cryptanalysis which mathematically extends rotational cryptanalysis to ARX ciphers with constants involved. In order to find distinguishers for ARX ciphers more efficiently, we study the automatic search techniques based on SAT/SMT solvers and a Python package SymPy, which automatically finds distinguishers in linear cryptanalysis and rotational-XOR cryptanalysis. The search tool is successfully applied to the block cipher family Speck, for which improved linear trails and rotational-XOR characteristics are found. For analysing ciphers with a substitution-permutation network structure, we study the block cipher LowMC which is designed for low multiplicative complexity. The S-box layer of LowMC has a relatively low algebraic degree, which makes it vulnerable to higher-order differential cryptanalysis. The round function allows us to adequately bound the number of monomials such that it leads to an efficient interpolation attack. As a result, the security claims of the original design are compromised. A third topic we study are the basic assumptions in differential cryptanalysis. By labelling the effective keys being the keys under which a characteristic has a nonzero probability, we propose an efficient algorithm to find singular characteristics that have a nonzero expected probability but with no effective keys, and find examples of singular characteristics in 5-round AES and Prince. Even though the assumptions enable the designers to estimate the security margin against differential cryptanalysis, the distinguishers found in this model might be invalid when the singular characteristics are not taken into consideration by the cryptanalyst.

Another topic in this thesis is the diffusion layer of block ciphers. We concentrated on the linear layers and the round constants in terms of the invariant subspace attack. It is shown that the dimension of possible invariant subspaces can be bounded under certain conditions, such that the invariant

subspaces in a round function will be destroyed by choosing the round constants. The design of new diffusion functions is currently receiving widespread attention. Although the diffusion functions in most ciphers are linear, some nonlinear functions may achieve the same diffusion effect while providing an extra layer of confusion. We proposed two types of nonlinear diffusion layers based on a nonlinear error-correcting code and T-functions, and rigorously analysed their cryptographic properties. By applying the nonlinear functions in toy ciphers, it hints that nonlinear diffusion layer is a promising building block for designing new lightweight block ciphers.

# Beknopte samenvatting

In dit proefschrift concentreren we ons vooral op cryptanalytische technieken voor blokcijfers, die ook voordelen bieden voor het ontwerp van cryptografische componenten.

Voor de ARX-structuur hebben we een nieuwe analysemethode voorgesteld, de rotationele-XOR-cryptanalyse, een wiskundige uitbreiding van rotationele cryptanalyse naar ARX-cijfers met constanten. Om onderscheidende elementen in de ARX-cijfers efficiënter te vinden, bestuderen we automatische zoektechnieken op basis van SAT/SMT-solvers en een Python-pakket SymPy, die beide automatisch onderscheidende elementen vinden voor lineaire cryptanalyse en rotationele-XOR cryptanalyse. Het zoekinstrument is met succes toegepast op de blokcijferfamilie Speck, waar verbeterde lineaire paden en rotationele-XOR-kenmerken werden gevonden. Voor de analyse van cijfers met een substitutie-permutatie-netwerkstructuur, bestuderen we het blokcijfer LowMC dat is ontworpen met het oog op lage multiplicatieve complexiteit. De S-box-laag van LowMC heeft een relatief lage algebraïsche graad die kwetsbaar is voor hogere-orde differentiële cryptanalyse, en de ronde-functie laat ons toe om adequaat het aantal eentermen te beperken, wat leidt tot een efficiënte interpolatie-aanval. Als gevolg hiervan komen de veiligheidsclaims van het oorspronkelijke ontwerp in het gedrang. Een derde onderwerp in cryptanalyse dat we hebben bestudeerd, zijn de basisaannames in differentiële cryptanalyse. Door aan te duiden dat de effectieve sleutels diegene zijn waaronder een karakteristieken een waarschijnlijkheid groter dan nul heeft, hebben we een efficiënt algoritme voorgesteld om singuliere karakteristieken te vinden, die een verwachte waarde groter dan nul hebben maar zonder effectieve sleutels, en hebben we voorbeelden van singuliere karakteristieken gevonden in 5 rounden van AES en Prince. Hoewel de aannames de ontwerpers in staat stellen om de beveiligingsmarge tegen differentiële cryptanalyse in te schatten, kunnen de onderscheidende elementen die in dit model worden gevonden ongeldig zijn als de singuliere karakteristieken niet door de cryptanalyst in overweging worden genomen.

Een ander aandachtspunt van dit proefschrift zijn de diffusielagen van blokcijfers. We concentreerden ons op de lineaire lagen en de rondeconstanten voor de invariante-deelruimteaanval. We tonen aan dat de dimensie van mogelijke invariante deelruimten onder bepaalde voorwaarden kan worden begrensd, zodat de invariante deelruimten in een ronde functie worden vernietigd door de rondeconstanten te kiezen. Het ontwerp van nieuwe diffusiefuncties is de laatste jaren een populair onderwerp. Hoewel de diffusiefuncties in de meeste cijfers lineair zijn, kunnen sommige niet-lineaire functies hetzelfde diffusie-effect bereiken terwijl ze een extra confusielaag bieden. We hebben twee soorten niet-lineaire diffusielagen voorgesteld op basis van een niet-lineaire foutcorrigerende code en T-functies, en hun cryptografische eigenschappen grondig geanalyseerd. Door de niet-lineaire functies in cijfers toe te passen, geeft het aan dat niet-lineaire diffusielaag een veelbelovende bouwsteen is voor het ontwerpen van nieuwe pluimgewicht blokcijfers.

# Abbreviations

| | |
|---|---|
| **AE** | Authenticated Encryption |
| **AES** | Advanced Encryption Standard |
| **ANF** | Algebraic Normal Form |
| **ARX** | Addition, Rotation, XOR |
| **CBC** | Cipher Block Chaining |
| **CNF** | Conjunctive Normal Form |
| **CP** | Constraint Programming |
| **DDT** | Difference Distribution Table |
| **DES** | Data Encryption Standard |
| **EDP** | Expected Differential Probability |
| **FPE** | Format-Preserving Encryption |
| **GFN** | Generalised Feistel Networks |
| **FFT** | Fast Fourier Transformation |
| **FHE** | Fully Homomorphic Encryption |
| **IoT** | Internet of Things |
| **IPsec** | Internet Protocol Security |
| **LAT** | Linear Approximation Table |
| **MAC** | Message Authentication Code |
| **MDS** | Maximum Distance Separable |

**MILP** Mixed-Integer Linear Programming

**MPC** Multi-Party Computation

**NMDS** Near-MDS

**SAT** Boolean Satisfiability Problem

**SMT** Satisfiability Modulo Theory

**SSH** Secure Shell

**SPN** Substitution-Permutation Network

**TLS** Transport Layer Security

**XOR** Exclusive Or

# List of Symbols

| | |
|---|---|
| $\oplus$ | XOR |
| $\boxplus$ | Modular addition |
| $1_{x \preceq y}$ | The symbol equals 1 when every bit of $x$ is no larger than that of $y$, otherwise it is 0 |
| $a$ | An element in a finite field |
| $\hat{a}$ | A vector over a finite field (when we stress that it is a vector) |
| $a_i$ | The $i$-th component of an element in a finite field $\mathbb{F}_{2^n}^m$ |
| $a_i[j]$ | The $j$-th bit of the $i$-th component of an element in a finite field $\mathbb{F}_{2^n}^m$ |
| $\boldsymbol{a}$ | An element of a ring |
| $\hat{\boldsymbol{a}}$ | A vector over a ring |
| $\mathbb{F}_{2^n}$ | A finite field of $2^n$ elements |
| Id | Identity operation |
| $\lvert S \rvert$ | The cardinality of a set $S$ |
| SHL | Left Shift by 1 |
| $wt(\cdot)$ | The Hamming weight |
| $\mathbb{Z}_{2^n}$ | The ring of integer modulo $2^n$ |

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Modern Cryptography

Since the early dawn of the information age, cryptography has played a vital role in almost every aspect of information security, and has had a profound impact on shaping the world. Even though words like "cryptocurrency" have gained a global-level attention in the past few years, cryptography is rather unfamiliar to the general public, either in the ancient times where cryptographic technology was controlled by the state or the military, or in the new century where cryptography has been deeply embedded in our daily lives. Nevertheless, violations of confidential information or personal data have become a common issue, through numerous hackings, phishings, frauds, malware or vulnerabilities in existing systems, which seem to be the inevitable by-product of the convenience and efficiency brought by the digital era. Therefore, the major goal of cryptography is to protect the confidentiality, authenticity, integrity and privacy of our data against malicious adversaries. Cryptographic algorithms, schemes and protocols have been widely deployed in communication systems, financial networks with online banking and payment, smart devices and tags evolving together with the rise of the *Internet of Things* (IoT), and so on.

Historically, the development of cryptography went through a long way until scientifically reliable solutions were eventually found. Modern cryptography began with the pioneering work of Shannon on the mathematical aspects of information theory, and later split into two directions: *symmetric-key (or secret-key) cryptography* and *asymmetric-key (or public-key) cryptography*. In either case, the keys are the source of protection. The same key is shared by both

parties in symmetric-key cryptography; while in public-key cryptography a private-public key-pair is distributed respectively for the participants in the communication channel. Generally speaking, symmetric-key algorithms have a better performance than the public-key ones in terms of speed and efficiency, whereas sharing the secret keys in symmetric-key cryptography is much more complicated than key distribution in public-key cryptography.

Unlike public-key cryptography, the security of symmetric-key algorithms is not based on the reduction to hard mathematical problems. Instead, they are based on the claim that no attacks can be found which are faster than exhaustive search. Therefore, before the adoption of new algorithms and schemes, especially for standards and official recommendations, they need to go through a comprehensive scrutiny by means of *cryptanalysis* or *attack*. To be specific, except for some legacy schemes, most of the cryptographic algorithms nowadays have to be publicly evaluated for several years before being accepted for practical applications. Nevertheless, major progress may still be achieved even in a highly-developed cryptosystem, and some might be devastating. For instance, the Heartbleed attack [81] revealed in the Transport Layer Security (TLS) protocol was one of the worst vulnerabilities found on the internet, and the KRACK attack [196] detected weaknesses in the Wi-Fi Protected Access (WPA2) protocol.

The theme of this thesis falls into the field of symmetric-key cryptography, especially on the cryptanalysis of symmetric primitives and the design of the components in a cipher.

## 1.2 Symmetric-key Primitives

Symmetric-key primitives are the major workhorses for almost all aspects of cryptography. They protect the confidentiality and the authenticity of information, where the latter includes data origin authentication and the integrity of the information. Meanwhile, symmetric-key primitives have contributed to many applications, for example, multi-party computation (MPC), fully homomorphic encryption (FHE), proof-of-work function in a blockchain, and the onion router in privacy technology. In this section, we will recall briefly the basic concepts of symmetric-key primitives.

### 1.2.1 Block Ciphers

Block ciphers are the fundamental components of symmetric-key primitives; they mainly focus on the functionality of encryption. In order to encrypt large files efficiently, the data are split into "blocks" of a fixed length, usually 64 or 128 bits. The secrecy comes with the secret key, which is shared between the communication parties.

**Definition 1.** *A block cipher with block size $n$ and key size $k$ is a tuple $(\mathcal{E}, \mathcal{D}, \mathcal{K})$, which is composed of an encryption function $\mathcal{E} : \mathbb{F}_{2^n} \times \mathbb{F}_{2^k} \to \mathbb{F}_{2^n}$, and a decryption function $\mathcal{D} : \mathbb{F}_{2^n} \times \mathbb{F}_{2^k} \to \mathbb{F}_{2^n}$. In addition, it satisfies that for a plaintext $p$ and a fixed key $K \in \mathcal{K}$, $\mathcal{D}(\mathcal{E}(p, K), K) = p$, i.e., a block cipher is a permutation under any given key.*

For efficiency reasons, block ciphers are often designed to be iterative functions, which are built from round functions with only a few simple operations.

**Definition 2** ([42])**.** *A block cipher is called an iterative block cipher with block size $n$, key size $k$, and $r$ rounds if it is composed of $r$ permutations for each key $K \in \mathbb{F}_{2^k}$:*

$$\mathcal{E}(K, \cdot) = f_{r-1}(K_{r-1}, \cdot) \circ f_{r-2}(K_{r-2}, \cdot) \cdots \circ f_0(K_0, \cdot),$$

*where $f_i : \mathbb{F}_{2^n} \times \mathbb{F}_{2^{k_i}} \to \mathbb{F}_{2^n}$ are the round functions, $K_i$'s are the round keys of $k_i$ bits generated by a key schedule $\mathcal{K} : \mathbb{F}_{2^k} \to \mathbb{F}_{2^{k_{r-1}}} \times \mathbb{F}_{2^{k_{r-2}}} \times \cdots \times \mathbb{F}_{2^{k_0}}$.*

Especially, if all the round functions are identical, it is called an *iterated block cipher*. Furthermore, if the round function $f(K, x)$ can be decomposed into $f'(x \oplus K)$, which means that the key is directly XORed into the state, then we call it a *key-alternating cipher* [61]. Many block ciphers are actually key-alternating ciphers, including the AES (Advanced Encryption Standard) [65].

Based on the specific round functions, block ciphers can be roughly classified into three types.

#### Feistel

One of the first Feistel ciphers is the former standard DES (Data Encryption Standard) published in 1977 [163]. A notable feature of the Feistel structure is that the round function only updates half (or part) of the entire state as shown in Figure 1.1. To be specific, the Feistel round maps $(x_L, x_R) \in \mathbb{F}_{2^n} \times \mathbb{F}_{2^n}$ to $(y_L, y_R) \in \mathbb{F}_{2^n} \times \mathbb{F}_{2^n}$ by

$$y_R = x_L \oplus F(x_R)\,,\ y_L = x_R\,,$$

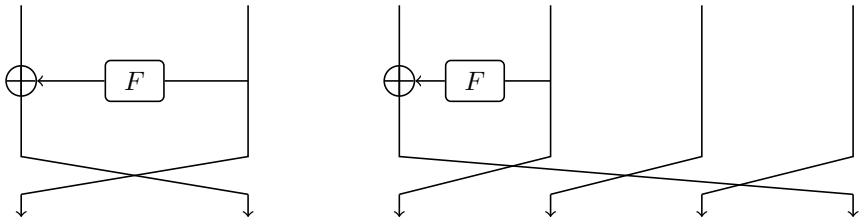



Figure 1.1: One round of a Feistel cipher.

where $F : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ is the round function. Note that the round function of a Feistel cipher is not necessarily bijective, and the decryption can reuse the encryption circuit to a large extent. When the two branches are of the same size, the Feistel network is called *balanced*; otherwise, it is *unbalanced*. As variants of the Feistel network, generalised Feistel structures have been proposed, including multiple branches (generalised Feistel networks, GFN); see Figure 1.1, and modular operations (also known as $\boxplus$-Feistel [38]).[1] Examples of Feistel ciphers include Misty [153], Kasumi [154], LBlock [208], Skipjack, Clefia [179], Piccolo [177], and Twine [190].

Apart from its practical importance, the Feistel structure acts as the theoretical bridge between pseudorandom functions and pseudorandom permutations, which is known as the Luby-Rackoff construction [151]. It was proven that a 3-round Feistel structure is sufficient to be a pseudorandom permutation, if the round function is a cryptographically secure pseudorandom function. There are a number of follow-up works on the theoretical security of Feistel structures, triggering many generic approaches for attacking Feistel ciphers, such as meet-in-the-middle and yoyo attacks [38, 80, 99].

The theoretical study of Feistel structures also finds applications in the so-called format-preserving encryption (FPE) [22] schemes which aim at enciphering plaintext over a small field meanwhile preserving the format. FPE somewhat resembles ciphertext stealing [61]. For instance, to encrypt a 16-digit credit card number (around 53 bits) with the AES, the plaintext needs to be padded to 128 bits, thus the ciphertext is longer than the plaintext and is probably not a 16-digit number anymore. Unbalanced Feistel structures [108, 158, 173], such as card shuffles, are extensively adopted in constructing provably secure FPEs. Meanwhile, FF1 and FF3 schemes for FPE have been standardised by NIST [82]; both are based on Feistel networks.

[1] Generalised Feistel structure take many different forms [109], where only one example is illustrated in Figure 1.1.

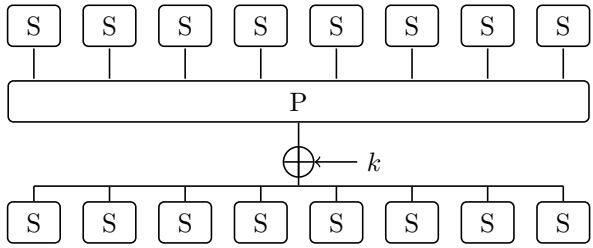



Figure 1.2: An illustration of the substitution-permutation network.

### Lai-Massey

The Lai-Massey scheme was developed for the IDEA cipher [133] by Lai and Massey. The state is split, and a round function (which is not necessarily invertible) is applied to the difference between the two parts of the state. Instead of XORs, modular additions and subtractions are used in the Lai-Massey schemes, under the design concept of "mixing operations from different algebraic groups".

### SPN

With DES being superseded by the AES, substitution-permutation networks (SPN) have taken over the majority of new block cipher designs. Usually, SPN ciphers are key-alternating, with the round key XORed at the end of each round. The round function of SPN ciphers is composed of a substitution layer of invertible S-boxes, and a permutation layer with linear functions, see Figure 1.2 for an illustration. The advantage of the SPN structure is that the components of an SPN cipher can be highly parametrised and flexible, by choosing appropriate S-boxes and linear functions. Moreover, SPN ciphers are often designed under the guidance of the Wide Trail Strategy [63] which provides provable bounds against several major cryptanalytic methods. We will discuss more detail in Section 1.3.3 and also in Chapter 5.

The S-boxes are often defined on 4 or 8 bits, and expected to achieve optimal differential uniformity and nonlinearity. For instance, the S-box of the AES is derived from the inverse function over the finite field $\mathbb{F}_{2^8}$, which is one of the best 8-bit S-boxes known to-date. Due to the tremendous number $(n!)$ of permutations on $n$ elements, it is generally difficult to find one with good cryptographic properties which also meets the implementation requirements. For 4-bit S-boxes, the properties and constructions have been extensively studied; for instance, it is known that there are 16 affine-equivalent classes [69, 138].

For the linear layer, there are various options, from bit-level permutations to word-level operations. The core component of many linear layer is carefully chosen such that it inherits good properties from the corresponding error-correcting codes.

**Modes of Operation.** A block cipher can only handle data with a fixed length. When utilising block ciphers in real-world applications, modes of operation [82] are a secure way to encipher data that is larger than the block size. A bad example of using a block cipher is the Electronic Codebook mode (ECB), where the ciphertext blocks are identical when the plaintext blocks are the same, thus it reveals some patterns in the plaintext. Cipher Block Chaining (CBC) is one of the most commonly-known modes for encryption. In CBC mode, the ciphertext of the previous encryption process is XORed to the plaintext of the current encryption, and the first block is processed with a random initialisation vector. The security of modes needs to be carefully deduced with provable security techniques by assuming that the underlying block cipher is a pseudorandom permutation. For an $n$-bit block cipher, CBC mode is unsafe if more than $2^{n/2}$ blocks are encrypted under the same key even though the block cipher itself has no cryptographic weakness. This is also known as the birthday bound. As a consequence, the block size of a block cipher cannot be too small. Although many 64-bit or 32-bit block ciphers are proposed for lightweightness, researchers warn for the possible vulnerabilities of some 64-bit block ciphers [33] in practice.

### 1.2.2 Stream Ciphers

Stream ciphers are a practical realisation of the idea of the one time pad. It generates a pseudorandom bit stream with a master key, an initialisation vector and some constants, and enciphers the plaintexts by XORing the key stream. Comparing with block ciphers, stream ciphers are fast and cheap in implementation, thus they are widely adopted in applications, such as SNOW 3G [83] in 3GPP, SALSA20 [24] in TLS by Google. In 2008, the EU project eStream selected a number of stream ciphers in a final portfolio for software- or hardware-oriented applications. Many of those are still of a great interest for academia and industry, such as SALSA20, GRAIN [105], and TRIVIUM [68]. Stream ciphers can be roughly classified into synchronising ciphers and self-synchronising ciphers, where the main difference is the dependency of the key streams on the ciphertexts.

One way to generate stream ciphers is to convert a block cipher using modes of operation. For instance, Cipher Feedback (CFB) mode is a self-synchronising stream cipher generated from a block cipher. Another option is to design stream ciphers from scratch. Unfortunately, there are no guidelines like the Wide Trail

Strategy in designing stream ciphers. Moreover, it is more difficult to analyse stream ciphers in general, due to their ad hoc structures.

### 1.2.3 Hash Functions

A hash function maps messages of arbitrary length into a digest of a fixed length $n$. The main functionality of hash functions is integrity and data authentication, which means one could easily check if the message is modified by checking the authenticity of the digest. There are three criteria to meet for a secure hash function $H(x)$:

- Pre-image resistance: For a given $y$, the complexity of finding an $x$ such that $H(x) = y$ is $2^n$.
- Second pre-image resistance: For a given $x$, the complexity of finding an $x'$ such that $H(x') = H(x)$ is $2^n$.
- Collision resistance: The complexity of finding a pair of $x$ and $x'$ such that $H(x) = H(x')$ is $2^{n/2}$.

Hash functions can be built based on block ciphers, using Davies-Meyer, Merkle-Damgård or Miyaguchi-Preneel constructions. The underlying block ciphers are often based on boolean operations and arithmetic. Examples include the SHA-1, SHA-2, MD hash families, and some finalists from the SHA-3 competition.

After the theoretical break of SHA-1 and the practical collision detected on MD5 in 2005, there was an urgent need in standardising a new secure and efficient hash function, like in the AES competition. In 2009, Keccak [30] won the SHA-3 competition, and became the new international standard for hash functions. Apart from Keccak, popular hash functions include SHA-256(-512) [86], Ripemd-160 [77], Blake2 [9], Skein [85], among them, SHA-256 and Ripemd-160 are used in several cryptocurrencies like the Bitcoin for transactions or proof-of-work.

### 1.2.4 MAC Algorithms

Message authentication codes, or MAC algorithms, are symmetric primitives for authentication and integrity, which have a similar role as digital signatures in public-key cryptography. Unlike many keyless hash functions, the communication parties using a MAC need to share a secret key, with which a tag is generated for the message. A secure MAC algorithm should be resistant

to forgery attacks, where the attacker forges tags to pass the verification process. MAC schemes are often based on block ciphers or permutations. For instance, CBC-MAC, is derived from the CBC mode with block ciphers, where the tag is generated as the encryption of the last ciphertext. HMAC [21] is another popular scheme for authentication, which is based on a hash function.

There are MACs without secrecy, which send plaintexts directly without encryption, see for instance the systematic authentication codes [147]. Authentication can also be combined with encryption, which is known as authenticated encryption (AE). For example, encrypt-and-MAC (E&M) produces the tag based on plaintext and only encrypts the plaintext, which is adopted in the Secure Shell protocol (SSH). Used in the SSL (Secure Sockets Layer) and TLS protocols, MAC-then-encrypt (MtE) generates the tag from the plaintext and then encrypts both. There is also encrypt-then-MAC (EtM) which first encrypts the plaintext then authenticates the ciphertext. EtM is utilised in the Internet Protocol security (IPsec).

Various attacks are discovered in MAC schemes due to the design itself or incorrect implementations, which severely compromise the data security. For instance, padding oracle attacks, or the Lucky Thirteen attack [1], take advantage of the padding errors in the plaintexts. In order to make authenticated encryption schemes secure, applicable, and robust, a competition CAESAR [54] calls for and evaluates new authenticated ciphers since 2013. The ciphers are expected to be secure, fast, and to offer advantages over the AES-GCM.

### 1.2.5 Permutation-based Primitives

The advantage of Keccak over other finalists in the SHA-3 competition is its novel design ideas and excellent performance in many platforms. It is a *permutation-based primitive* [28, 29] with 5-bit S-boxes and strong diffusion operations. Keccak adopts a *sponge structure* (see Figure 1.3) as the compression function which is both efficient and secure. The popularity of Keccak led to the study of permutation-based primitives. Block ciphers can be used to generate other primitives by employing modes of operation, while permutations can be used in a sponge structure to encrypt and/or authenticate messages. In a sponge structure, the function $f$ is a cryptographically-secure permutation. The state size of $f$ is called the width, which is split into the rate $r$ and the capacity $c$ indicating the efficiency and security strength, respectively.

To process a message, an input of variable lengths is absorbed into the sponge. Then, it squeezes out the outputs of an arbitrary length with the fixed-length permutation in an iterated construction. The sponge structure was later extended into the duplex structure, where both sponge and duplex have keyed

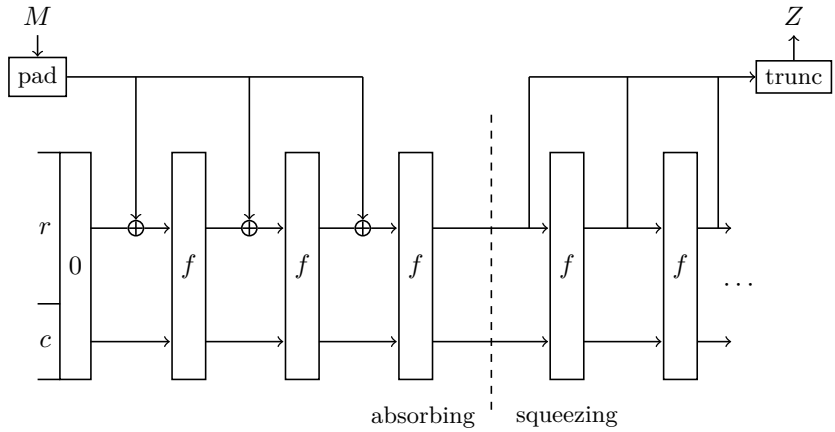



Figure 1.3: The sponge structure.

variants, namely the donkey sponge and the monkey duplex [28]. Descendants of KECCAK are proposed, such as the authenticated encryption schemes KEYAK [32] and KETJE [31], a fast hashing function KANGAROOTWELVE [27], and a pseudorandom function KRAVATTE [26]. Meanwhile, the sponge construction has also been adopted in several lightweight hash functions such as PHOTON [101], SPONGENT [43] and QUARK [8].

### 1.2.6 Lightweight Symmetric Cryptography

With the popularity of the IoT, small devices from sensors to RFID tags are able to be connected and communicate with each other through networks. With billions of such small devices, the security and privacy of the information may be at stake due to various types of adversaries, where some might be the users themselves. Thus the information should be properly protected by cryptographic schemes.

Most cryptographic algorithms are designed with a strong guarantee on security as the main priority. Even though there are many studies on highly optimised implementations, especially for the AES, the algorithms may not be suitable or efficient enough in constrained software or hardware devices. Such devices have restrictions on the memory, area, throughput, energy or power consumption, and some are based on microprocessors and micro-controllers. As a result, instead of tweaking the implementations for current algorithms to the limit, new symmetric-key primitives are designed to fit constrained environments, with novel structures and building components proposed. The challenge is to

find the right trade-off among security margin, efficiency and cost. Despite the insignificant size of the devices, lightweight cryptography is by no means weak cryptography. The primitives are still expected to be secure against adversaries, even though the attackers in this context may possess less computational power or only have restricted access to the plaintexts or ciphertexts.

During the past decade, a large number of lightweight primitives have been proposed by both academia and industry, see for instance a comprehensive survey by Biryukov *et al.* [39]. Among them, one of the first lightweight block ciphers is PRESENT, which was presented in CHES 2007 and later became an ISO/IEC standard in 2012 along with CLEFIA [179]. Afterwards, the community witnessed a boost in new block cipher proposals, including PRINCE [47], KLEIN [93], LED [102], RECTANGLE [212], PRIDE [5], PRINTCIPHER [125], SIMON and SPECK [16], SKINNY [19], GIFT [13] and so on.

Lightweight stream ciphers, hash functions and authenticated encryption schemes can be built upon lightweight block ciphers. Nevertheless, there are new proposals such as PHOTON [101], SPONGENT [43], HARAKA v2 [129], KREYVIUM [55], LIZARD [104], as well as many CAESAR candidates [31, 207].

## 1.3 Cryptanalytic Techniques

### 1.3.1 Models of Cryptanalysis

Designs and cryptanalysis evolve with dynamic interactions. For the two parties in the game, a basic agreement is *Kerckhoffs' Principle*, which states that the entire cryptosystem, except for the secret key, is publicly known by the adversary. The secrecy of the secret key is so vital, that specific standards and regulations are formulated on the generation, management and storage of the keys. The focus of this thesis is the security of cryptographic algorithms when the secret key is correctly deployed in the system. In this section, we introduce the classifications of attacks based on the power of the adversary. We will mainly focus on the cryptanalysis of block ciphers for simplicity, where the terminology and methodology may also be applicable to other types of primitives.

Ideally, a cryptographic system is secure even when the attacker has unlimited computing power, which is known as *information-theoretical security* or *unconditional security*. In other words, the ciphertext will not leak any information about the plaintext, except the length, no matter how powerful the adversary is. One example that offers such secrecy is the one-time pad, where the random key-string completely destroys the relation between plaintexts and

ciphertexts. However, the scheme is impractical as the length of the secret key should be at least the same as that of the message. Meanwhile for real-life attackers, with either a supercomputer or a laptop, the computational power is limited. Under such a model, the security of cryptographic schemes we consider is called *computational security*. In other words, no current technology could break the system. However, it might become vulnerable in the future. The RSA cryptosystem is an example. Even though it is safe and widely applied nowadays, the threat from quantum computers is the sword of Damocles, which in return leads to an active research area, known as the post-quantum cryptography [25].[2]

In the context of symmetric-key cryptography, computational security may be violated in three models, taking the amount of exposed information into consideration.

- Black-box Model. A cryptographic algorithm is considered as a black box, where the adversary only has access to the plaintexts and/or the ciphertexts, but has no control nor information on the intermediate values. In this model, the mathematical properties of the algorithm are closely examined to exclude design flaws and weaknesses. This is what a cryptanalysis means in a narrow sense. Any algorithm should at least be secure in the black-box model.
- Grey-box Model. Cryptographic algorithms are implemented on software or hardware platforms. As a result, a mathematically secure cipher might be compromised because of incorrect implementations, where the physical characteristics of the implementations can be used to recover the secret key. Attacks in grey-box model are also called implementation attacks. They include for example, timing attacks, acoustic attacks, cache attacks, differential fault attacks, power analysis attacks, and so on.
- White-box Model. White-box adversaries, which are the most powerful attackers, have access to the implementation details as well as the dynamic execution. For instance, an attacker is able to see or tamper with the secret key in the memory.

This thesis mainly deals with adversaries in black-box model. More specifically, the mathematical aspects related to the design and analysis of block ciphers, which includes the following attacking goals.

- Key Recovery Attacks. In the one-time pad, the secret key is random and has at least the same length as the message. For practical applications,

[2]Dedicated research on quantum cryptanalysis of symmetric primitives can also be found in a couple of research papers [113, 114].

the secret key is generated from a master key, which is of a fixed length. A straightforward approach to recover the key is through an exhaustive key search. For a $\kappa$-bit key, the complexity of exhaustive search is around $2^{\kappa-1}$ on average. Therefore, the key length cannot be too small. The recommended key length by NIST [14] is at least 112 bits for the federal government; while for top-secret level documents, the key length should be 192 or 256 bits for long-term security. As we will discuss in the following chapters, the attacker might take advantage of the design details to mount key recovery attacks with complexity lower than exhaustive search. This is called breaking an algorithm.

- Distinguishing Attacks. Instead of recovering the key, an attacker may also try to distinguish a cipher from random. Although not all distinguishing properties could be efficiently converted into a key recovery attack, a distinguisher is often essential in launching an efficient attack.

Adversaries can be either passive or active, from simply observing the ciphertexts to deliberately asking for the encryption of chosen plaintexts. In reality, one should be prudent on the assumptions about the adversary and always prepare for the worst when it comes to security. For instance, deliberate actions were taken in order to lure the opponent to encrypt messages with certain information which were successfully applied in World War Ⅱ. Based on the access to the plaintexts and ciphertexts, the attacks can be classified into the following categories.

- Ciphertext-only attack. The adversary is completely passive, and only has information about the ciphertext and statistical information on plaintexts. A good cipher is supposed to generate indistinguishable ciphertexts, which leaks no information about the plaintexts.
- Known-plaintext attack (KPA). The adversary knows some plaintext-ciphertext pairs, which might be advantageous for deducing the secret key.
- Chosen-plaintext attack (CPA). The adversary has access to the encryption oracle, and is able to ask for the encryption of the plaintexts at his/her will.
- Chosen-plaintext chosen-ciphertext attack (CCA). The adversary has access to both encryption and decryption oracles, and may (adaptively) ask for the decryption of the chosen ciphertexts.

Parallel to the plaintexts and ciphertexts, assumptions can be made about the keys as well. In principle, the secrecy of the keys should be protected. However,

this is possibly violated due to many practical reasons. For instance, the update process of the secret key is not random enough, such that some relations can be found between two master keys used in a system. Besides, it is interesting for researchers to study the security of cryptographic algorithms given certain conditions on the keys, which may lead to better insights on the designs and analysis.

- Single-key model. This is the default attack model, where the key is a fixed but unknown value.
- Open-key models.
  - Related-key model. The adversary may query the encryption or decryption oracle under a pair of keys with a certain relation, for instance, the XOR difference of the keys is fixed and known.
  - Weak-key model. The adversary is able to chose the secret key from a set of weak keys.
  - Known-key model. Under such a model, the adversary knows the secret key, based on which the adversary is able to deduce properties of the cipher. The attack is often of a distinguishing nature, which is of a theoretical significance in understanding and analysing primitives [126].

### 1.3.2 The Toolbox of a Cryptanalyst

Beginning with the invention of differential cryptanalysis and many other techniques, a formalised research field has been established on the systematic analysis of symmetric primitives, with a large number of cryptanalytic techniques developed. As the elemental components in many primitives are block ciphers, most of the techniques are studied based on block ciphers; they may also be further converted to finding collisions in hash functions or forging tags in MAC algorithms. As we have mentioned before, a good distinguisher is essential for a successful attack. The challenging task is to figure out which is applicable and how to find one. Based on the certainty of the distinguishers, most of the cryptanalytic methods can be roughly categorised into cryptanalysis with a statistical distinguisher or a deterministic distinguisher. For simplicity, we call them *statistical cryptanalysis* and *deterministic cryptanalysis*, respectively. In statistical cryptanalysis, the distinguisher exists with a certain probability. Usually, the statistical relations between the plaintexts and ciphertexts are studied, where the probability is determined by the cipher as well as the secret key. Meanwhile, a distinguisher in deterministic cryptanalysis is often

Table 1.1: An overview on a fraction of main cryptanalytic techniques

| Statistical distinguisher | Deterministic distinguisher |
|---|---|
| Differential cryptanalysis* | Higher-order differential cryptanalysis† |
| Linear cryptanalysis† | Impossible differential cryptanalysis* |
| Truncated differential cryptanalysis* | Integral cryptanalysis† |
| Differential-linear cryptanalysis* | Zero-correlation linear cryptanalysis† |
| Boomerang attack† | Division property† |
| Rectangle attack† | Interpolation attack† |
| Multiple linear cryptanalysis† | Zero-sum distinguisher† |
| Multiple differential cryptanalysis* | Algebraic attack† |
| Multidimensional linear cryptanalysis† | |
| Rotational cryptanalysis* | |
| Rotational-XOR cryptanalysis* | |
| Polytopic cryptanalysis* | |
| * - Known Plaintext Attack, | †- Chosen Plaintext(/Ciphertext) Attack |

independent of the secret key, and derived based on the algebraic properties of a cipher. We summarise the major cryptanalytic techniques [7, 11, 34, 35, 36, 45, 56, 106, 107, 111, 122, 124, 131, 135, 152, 191, 198] in Table 1.1, where we assume that the attacker works in the single-key model. Note that it is possible to find some corresponding distinguishers in the open-key model as well. For example, related-key differential cryptanalysis finds differential characteristics under a pair of related keys having a certain difference.

Note that Table 1.1 covers only a fraction of all distinguishers. For instance, an invariant-subspace distinguisher [136] holds with probability 1 for a weak-key class, thus it belongs to the deterministic cryptanalysis column but only under a weak-key model. Similar exceptions include the nonlinear invariant attack [193] and subspace-trail cryptanalysis [94], where the former works only for weak keys and the latter can be regarded as a generalisation of integral cryptanalysis and truncated differential cryptanalysis.

In general, it is difficult to find a good distinguisher. Typically, many human hours were spent to search for an optimal distinguisher, which requires delicate inspection of the mathematical properties as well as sharp observations based on one's expertise. In recent years, computer-aided search methods have been successfully applied in finding distinguishers automatically, which dramatically improves the efficiency in cryptanalysis as well as in designs. Another good

news is that the distinguishers are not isolated. It is proven that differential cryptanalysis and linear cryptanalysis can be linked through a complicated relation in the Fourier domain [58], which however is rather a theoretical observation than predicting new distinguishers from one another. For impossible differential cryptanalysis, zero-correlation linear cryptanalysis and integral cryptanalysis, connections can be found for ciphers within a class where the linear layers satisfy certain conditions [187]. The observations benefit the theoretical predictions on the length of the distinguishers [186], or yield a new distinguisher based on the knowledge of a known one of another type.

In order to complete an attack, techniques for key recovery, collision finding or tag forgery need to be considered in detail. Examples include meet-in-the-middle attack, rebound attack [156], cube attack [76] and guess-and-determine attack. To measure the effectiveness of an attack, one needs to consider the complexities in terms of data, time and memory. Usually, the data complexity is determined by the strength of a distinguisher. Time complexity and memory complexity can be finely adjusted by a trade-off, depending on the computational resource possessed by an attacker, which is known as the time-memory trade-off attack.

### 1.3.3 Provable Resistance Against Certain Attacks

During the early practice of designing symmetric primitives, the design philosophy was often based on rules of thumb, which may overlook some possible flaws. The Wide Trail Strategy [61], adopted by Daemen and Rijmen in their milestone proposal of Rijndael [65], is a successful approach to guide the design of block ciphers. Here, we recall the strategy in the context of differential cryptanalysis.

In an SPN cipher, an S-box is called active when it has a nonzero input difference. The Wide Trail Strategy bridges the provable bounds on the probability of characteristics with the minimum number of active S-boxes in a cipher, where the branch number [61] of the linear layer plays a significant role.[3]

**Definition 3.** *Let $s, n$ be integers. The differential branch number of a linear function $f : \mathbb{F}_{2^s}^n \to \mathbb{F}_{2^s}^n$ is given by*

$$B_d(f) = \min_{a \neq 0} \{wt(f(a)) + wt(a)\}.$$

*Denote the matrix corresponding to the linear function by $M$. When $M$ is a maximum distance separable (MDS) matrix, $B_d(f) = n + 1$, which is optimal.*

[3]The definition of branch number here can also be extended to nonlinear functions, which will be shown in Chapter 5.

*If $M$ is sub-optimal, then $B_d(f) = n$. In such a case, the matrix is called Near-MDS or NMDS.*

It can be shown that $2 \leq B_d(f) \leq n+1$ when $f$ is invertible. Diffusion functions with branch number $B_d(f) \leq n$ have become popular in designing lightweight block ciphers; for instance, PRINCE [47] and MIDORI [12] utilise functions with branch number 4, while PRESENT [44] adopts a bit-shuffling function with branch number 2.

It is possible to mathematically deduce the minimum number of active S-boxes of a cipher based on the branch number, while the bound can also be obtained by running a program as we will discuss in detail in Chapter 2.

## 1.4 Thesis Outline

The main focus of this thesis is to study block ciphers from the viewpoint of a cryptanalyst, as well as to deduce the consequences and benefits for designing new ciphers. In the next chapters, we will dig into the cryptanalytic techniques for block ciphers, and discuss in depth the systematic framework of finding a distinguisher, the attacks on specific ciphers or structures, and also novel approaches to construct new components.

The organisation of the rest of this thesis is as follows.

We start with finding a good distinguisher for block ciphers in Chapter 2, especially ciphers with an ARX structure. After a comprehensive overview of the basic cryptanalytic techniques for ARX ciphers, we propose a new attack called rotational-XOR cryptanalysis, which studies a novel statistical behaviour of ARX ciphers when rotational cryptanalysis is inapplicable. In order to efficiently search for distinguishers, even for ciphers with relatively large block size, we focus on the automatic search techniques on ARX ciphers based on SAT/SMT solvers. Due to the combinations of simple operations in ARX, we are able to develop a tool based on Python, which automatically generates the SAT/SMT files for searching a certain type of distinguishers and further automates the process of computer-aided cryptanalysis. We choose the SPECK family as an example, and find optimal linear trails and rotational-XOR characteristics for round-reduced versions of all instances in the family, which also improves previous results in the literature. The automatic search technique can be beneficial for the designers, since the circle of design and evaluation can be largely shortened for future ARX ciphers. The content of this chapter was published in the following papers [7, 71, 146, 150, 171].

In Chapter 3, the cipher LowMC aiming for low multiplication complexity is analysed, with some of the security claims broken. LowMC is a block cipher with special optimisations for the applications in MPC and FHE. It took a rather novel design philosophy, adopting a partial S-box layer with a dense randomised linear layer. However, the lightweight nonlinear layer allows one to explore the higher-order differential property which penetrates a relatively large number of rounds. Meanwhile, optimised interpolation attacks can be launched to recover the secret key by solving a linear equation system. The time complexity of the attack depends on the number of unknown variables. In order to estimate the parameter with more accuracy, a novel method is proposed which combines the previous two key-recovery approaches in a dynamic way. As a response, the designers have revised the original design to prevent such attacks. This chapter is based on a published paper [75].

In Chapter 4 and Chapter 5, we study the diffusion layers of block ciphers, especially SPN ciphers. While the S-box operations often receive much attention in research, diffusion layers plays a crucial and irreplaceable role in the design and cryptanalysis as well. In Chapter 4, we study the influence of linear layers on the invariant subspace attack. By extracting properties of the linear layer, we are able to show a bound on the dimension of possible invariant-subspace distinguishers. In Chapter 5, we focus on designing new diffusion layers which are in fact nonlinear. Under the guidance of the Wide Trail Strategy, a nonlinear diffusion layer with good diffusion parameters and decent nonlinear properties may allow a block cipher to achieve resistance against differential and linear cryptanalysis within a smaller number of rounds. We propose two types of nonlinear diffusion functions, with their cryptographic properties analysed in depth. As an application, toy ciphers are constructed that show a stronger resistance compared with ciphers of a similar structure but with a linear diffusion layer. Chapter 4 is based on a publication in [148]. The content of Chapter 5 will appear in a journal [149].

For most of the statistical cryptanalytic techniques, the probability of a distinguisher is estimated under certain assumptions on the round function and round keys, which might not necessarily be true in reality. In Chapter 6, we focus on one of the most significant questions in differential cryptanalysis, namely, the accuracy of the probability estimation for a differential characteristic. We propose the concept of effective keys, and they are the keys for which a characteristic has a nonzero probability. As a result, it is possible that some characteristics have no effective keys at all, which we call *singular characteristics*. An algorithm is proposed in Chapter 6, as well as concrete examples of singular characteristics in practical ciphers.

Finally, in Chapter 7, we conclude the thesis with a few remarks, and discuss future work.

# Chapter 2

# Automatic Search Techniques on ARX

In this chapter, we study the automatic search techniques on ARX ciphers, and their applications in cryptanalysis. We start by recalling the basics of ARX ciphers and previous cryptanalytic techniques in Section 2.1 and 2.2. Rotational-XOR cryptanalysis is introduced in Section 2.3 which is a novel cryptanalytic method on ARX. Afterwards, a general framework for automatic search techniques is discussed in Section 2.4. The automatic search techniques on ARX can be packaged into an automatic tool, which only costs a minimal effort of an cryptanalyst in analysing ARX ciphers. We apply the automatic tool to the block cipher family SPECK in Section 2.5. Finally, we conclude in Section 2.6. The contents on rotational-XOR cryptanalysis and its applications to SPECK in this thesis are based on the collaborative works with Ashur, De Witte, and Ranea, see [7, 71, 146, 171]. The main contributions in this thesis include developing the automatic search techniques with SAT/SMT solver for ARX ciphers, and the mathematical model for Rotational-XOR cryptanalysis.

## 2.1 ARX Primitives

ARX, which stands for modular Addition, Rotation and XOR, is a widely used design philosophy. Through the interactions of simple operations, ARX provides strong cryptographic properties. Meanwhile, the operations can be efficiently implemented in software, so that it is favoured by some designers and a number of ARX-based designs were proposed. Among them there are the hash

functions BLAKE [9] and SKEIN [85], which are two of the five SHA-3 finalists, stream ciphers such as SALSA20 [24] and CHACHA [23], block ciphers such as TEA [205], XTEA [164], HIGHT [110] and SPECK [16], and the MAC algorithm CHASKEY [159]. In 2016, the ARX-based primitives SPARX and LAX [73] were designed with provable resistance against differential and linear cryptanalysis. Different from the classification that we have recalled in Chapter 1, ARX primitives are not necessarily designed with one of those structures, instead they often take ad hoc design approaches.

As an extension of ARX, a structure replacing the addition with the bitwise AND, is of special interest because the bitwise AND simplifies cryptanalysis and improves hardware efficiency. It gained attention and popularity with the SHA-3 hash function KECCAK [30], the NSA block cipher SIMON [16], and the AE scheme NORX [10].

### 2.1.1 SPECK Family of Block Ciphers

In this section, we introduce the ARX cipher SPECK which we will extensively study; it serves as a proof-of-concept to the methodology we propose for general ARX primitives in this chapter.

SPECK is a family of lightweight block ciphers designed by the NSA in 2013 [16]. A member of the family is denoted by SPECK$2n/mn$, where the block size is $2n$ for $n \in \{16, 24, 32, 48, 64\}$, and the key size is $mn$ for $m \in \{2, 3, 4\}$.

The round function of SPECK receives two words $x^i$ and $y^i$, and a round key $k^i$, all of size $n$, and outputs two words of size $n$: $x^{i+1}$ and $y^{i+1}$, such that

$$(x^{i+1}, y^{i+1}) = F_{k^i}(x^i, y^i) = (f_{k^i}(x^i, y^i), f_{k^i}(x^i, y^i) \oplus (y^i \lll \beta)),$$

where

$$f_{k^i}(x^i, y^i) = ((x^i \ggg \alpha) \boxplus y^i) \oplus k^i .$$

The SPECK key schedule algorithm uses the same round function to generate the round keys. Let $K = (l^{m-2}, ..., l^0, k^0)$ be a master key for SPECK$2n$, where $l^i, k^0 \in \mathbb{F}_{2^n}$. The sequence of round keys $k^i$ is generated as

$$k^{i+1} = f_{ct}(l^i, k^i) \oplus (k^i \lll \beta),$$

for

$$l^{i+m-1} = f_{ct}(l^i, k^i),$$

with $ct = i$ the round number starting from 0.

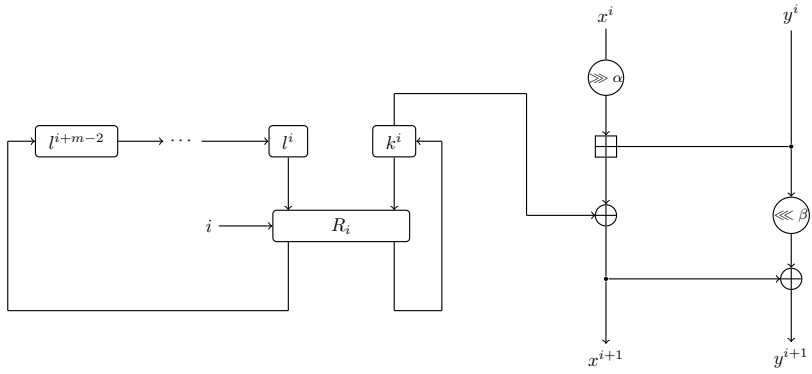



Figure 2.1: One round of SPECK.

The rotation offset $(\alpha, \beta)$ is $(7, 2)$ for SPECK32, and $(8, 3)$ for the larger versions. A single round of SPECK with $m = 4$ is depicted in Figure 2.1. For more details, we refer to the original design [16] and a note by the designers [17].

## 2.2 Previous Cryptanalytic Techniques

Despite the simplicity of ARX designs, it is difficult to find their cryptographic weaknesses. Indeed, if the structure is not appropriately applied, for instance without rotation, the system easily becomes vulnerable [169]. However in general, it is not easy to break an ARX cipher, and the cryptanalytic tools applicable to ARX primitives are rather limited. In this section, we will recall some cryptanalysis techniques: Section 2.2.1, 2.2.2 and 2.2.3 recap differential cryptanalysis, linear cryptanalysis and their variants, which are generally effective for most symmetric-key primitives. In Section 2.2.4, we give a brief introduction to rotational cryptanalysis.

### 2.2.1 Differential Cryptanalysis

Differential cryptanalysis is a statistical attack on symmetric-key primitives proposed by Biham and Shamir in 1990 [36]. It theoretically breaks the security of DES, and is one of the first techniques that can be applied to most modern ciphers. In a nutshell, an attacker encrypts plaintexts with a fixed input difference, and analyses the differences in the outputs. For a random permutation, any difference might occur, while a cipher may possess weaknesses

such that some output difference is more probable than others, which leads to a distinguisher.

**Definition 4** ([132])**.** *Let $(G, \otimes)$ be an Abelian group. The difference between two elements $X$ and $X^*$ is defined as $X + X^{*-1}$, where $+$ is the group operation and $X^{*-1}$ is the inverse element of $X^*$ in $G$.*

For finite fields with characteristic 2, the difference is often defined with the XOR operation: $X \oplus X^*$. The differential probability of a vectorial Boolean function is defined as follows.

**Definition 5.** *Let $\delta \in \mathbb{F}_{2^n}$ be the input difference and $\Delta \in \mathbb{F}_{2^m}$ be the output difference. For a vectorial Boolean function $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^m}$, the differential probability of $\delta \to \Delta$ is defined as*

$$dp(\delta \to \Delta) \triangleq \frac{\#\{x \in \mathbb{F}_{2^n} | f(x) \oplus f(x \oplus \delta) = \Delta\}}{2^n}.$$

A table that contains the number of right pairs for all input and output differences is called a difference distribution table (DDT). For functions with a small input size, for instance an S-box, the DDT can be efficiently generated. However, it is currently infeasible to obtain the DDT of a block cipher, considering its large block size and the fact that every master key instantiates a different permutation. To facilitate the analysis, recall that block ciphers are often iterative functions, we define a *differential characteristic* as below.

**Definition 6.** *For an iterative function $f = f_{r-1} \circ \cdots \circ f_1 \circ f_0$, a sequence of differences*

$$\Omega : \delta_0 \xrightarrow{f_0} \delta_1 \xrightarrow{f_1} \delta_2 \to \cdots \to \delta_{r-1} \xrightarrow{f_{r-1}} \delta_r$$

*is called an $r$-round differential characteristic of $f$.*

A differential $(\delta \to \Delta)$ over $f = f_{r-1} \circ \cdots \circ f_1 \circ f_0$ contains all differential characteristics with $\delta_0 = \delta$, and $\delta_r = \Delta$, where the probability is the sum on that of its characteristics. In general, there is no efficient way to precisely obtain the probability of a differential characteristics. To this end, we introduce some fundamental assumptions on block ciphers.

### Markov Cipher

Most block ciphers are designed as an iterative function with simple rounds. It is possible that the rounds are mutually dependent so that there is no simple formula to compute the overall probability of a characteristic. To simplify the

analysis, certain hypotheses are assmued as the foundation of many cryptanalytic techniques, which have been formulated by Lai, Massey and Murphy [132]. Here, we follow the interpretations in the AES design [65].

- **Markov cipher.** A Markov cipher is an iterative cipher whose round function satisfies the condition that the differential probability is independent of the choice of one of the component plaintexts under an appropriate definition of difference.
- **Hypothesis of stochastic equivalence.** For virtually all values of the cipher key, the probability of a differential characteristic can be approximated by the expected value of the probability of the differential characteristic, averaged over all possible values of the cipher key.
- **Hypothesis of independent round keys.** The round keys of a cipher are derived from the cipher key by means of the key schedule. The hypothesis states that the expected probability of a differential characteristic averaged over all possible values of the cipher key, can be approximated by the expected probability of the differential characteristic, averaged over all independently specified round keys values.

Over decades of practice, it is widely accepted that the assumptions are plausible and close to the experimental reality for many ciphers, meanwhile some discrepancies have been observed in analyses as well [103, 123, 200]. In this chapter, our analyses will be based on these assumptions and verified by experiment. In Chapter 6 of this thesis, we will study in depth the hypotheses and their influence on cryptanalysis.

### Key Recovery Techniques

Differentials and characteristics only distinguish a block cipher from random permutations, so further techniques for recovering the secret key are required. Here, we describe a basic approach for the recovery of the last-round key, which is the so-called 1R-attack. The attacker has the knowledge of an $r$-round differential distinguisher $(a \to b)$ of a block cipher. After encrypting the plaintext pairs with input difference $a$ over $r+1$ rounds and obtaining the ciphertexts, the attacker guesses some necessary key bits in the $(r+1)$-th round key, and executes a partial decryption on the ciphertexts. If the output difference $b$ does not occur with the predicted probability, then the guessed key is filtered out as a wrong guess. At the end of this process, the attacker obtains *suggested key values*. Note that wrong key values may be suggested due to the coincidence with wrong pairs. The signal-to-noise ratio $S/N$ is defined as the

ratio between the number of right pairs and the average count of the incorrect subkeys in a counting scheme. It was shown in [36] that the signal-to-noise ratio is proportional to the probability $p$ of the differential distinguisher, thus the data requirement for the attack to succeed is approximately $c \cdot p^{-1}$, where $c$ is a constant that depends on $S/N$.

#### Differential Cryptanalysis on ARX

With alternative definitions of the differences, differential cryptanalysis of the ARX ciphers has two versions: with XOR difference where $X^* = X \oplus \delta$ and additive difference where $X^* = X \boxplus \delta$ [145].

**Definition 7.** *Let $x, y$ be elements in $\mathbb{F}_{2^n}$, and $\delta_x, \delta_y, \delta_z \in \mathbb{F}_{2^n}$ be the difference values. Then the XOR differential probability of addition is defined as*

$$xdp^+(\delta_x, \delta_y \to \delta_z) = \Pr[(x \oplus \delta_x) \boxplus (y \oplus \delta_y) = (x \boxplus y) \oplus \delta_z] .$$

*Similarly, the additive differential probability of XOR is*

$$adp^\oplus(\delta_x, \delta_y \to \delta_z) = \Pr[(x \boxplus \delta_x) \oplus (y \boxplus \delta_y) = (x \oplus y) \boxplus \delta_z] .$$

Formulae for computing the differential probability $xdp^+$ and $adp^\oplus$ were studied in a number of papers [144, 145, 160], based on the observation that the modular addition is a T-function [120] (see the definition in Appendix A). As a result, the theoretical probability of the differential characteristics can be obtained efficiently.

### 2.2.2 Linear Cryptanalysis

Linear cryptanalysis [152] was proposed by Matsui in early 1990's; it is the first experimental cryptanalysis against the DES. Linear cryptanalysis investigates linear relations among the plaintext, ciphertext and the secret key.

**Definition 8.** *Let $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^m}$ be a vectorial Boolean function. Assume that the masks for input $x$ and output $f(x)$ are $\Gamma_{in}$ and $\Gamma_{out}$. The correlation of the linear approximation is defined as*

$$C(\Gamma_{in}, \Gamma_{out}) = 2 \cdot \Pr(\Gamma_{in} \cdot x \oplus \Gamma_{out} \cdot f(x) = 0) - 1 .$$

Equivalently, the correlation can also be written as a Walsh transformation,

$$C(\Gamma_{in}, \Gamma_{out}) = 2^{-n} \sum_{x \in \mathbb{F}_{2^n}} (-1)^{\Gamma_{in} \cdot x \oplus \Gamma_{out} \cdot f(x)} .$$

The square of the correlation $C^2$ is sometimes called the *linear probability* of an approximation, which is analogous to the probability in differential cryptanalysis.

In some analysis, bias is used instead of correlation, which is defined as $\epsilon = C(\Gamma_{in}, \Gamma_{out})/2$. We will stick to correlations in the rest of this thesis, and the notations mostly follow the analysis of correlation matrices by Daemen *et al.* [62]. Similar to differential cryptanalysis, we consider the linear approximation of an iterated function. Let $f = f_{r-1} \circ \cdots \circ f_1 \circ f_0$ be an iterated function. Linear approximations $(\gamma_i, \gamma_{i+1})$ of a single round $f_i$ can be concatenated into a *linear trail* $(\gamma_0, \gamma_1, \cdots, \gamma_r)$ of $f$.[1] In order to estimate the correlation of a linear trail, the following lemma is often used.

**Lemma 1** ([62]). *Let $(\gamma_0, \gamma_1, \cdots, \gamma_r)$ be a linear trail of an iterated permutation. Then, the correlation of the linear trail can be calculated as*

$$C(\gamma_0, \gamma_r) = \prod_{i=0}^{r-1} C(\gamma_i, \gamma_{i+1}).$$

A linear approximation $(\Gamma_{in}, \Gamma_{out})$ of a block cipher is called a linear hull [167], which contains all linear trails with masks $\Gamma_{in}$ and $\Gamma_{out}$. The correlation of a linear hull is the sum of the correlations of all the linear trails as predicted by the correlation matrix [62], however, sometimes the exact value is difficult to obtain. Hence, the *potential* (average squared correlation) of a linear hull is defined as

$$ALP(\Gamma_{in}, \Gamma_{out}) = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} C(\Gamma_{in}, \Gamma_{out})^2.$$

### Key Recovery Attack

The basic key recovery technique in linear cryptanalysis is Matsui's Algorithm 2 [152], which is based on the *wrong key randomisation hypothesis*. In short, it is assumed that with a wrongly guessed key, the partial decryption does not "peel off" the last round and leads to some random behaviour.[2] With a linear distinguisher of correlation $c$, the data complexity is estimated to be proportional to $c^{-2}$ up to some constant.

[1] A linear trail is sometimes called a linear characteristic [152].

[2] A number of follow-up studies (see for instance [6, 46]) finely adjust the original hypothesis of wrong key randomisation, which leads to a more accurate estimation of the data complexity and the success probability.

### 2.2.3 Variants of Differential and Linear Cryptanalysis

Variants of differential cryptanalysis were proposed, including truncated differential cryptanalysis [124], higher-order differential cryptanalysis [124, 131], impossible differential cryptanalysis [34, 121], boomerang attack [198], rectangle attack [35], etc. Many of the variants explore the combined effect of characteristics [56, 134, 194, 201].

Linear cryptanalysis has been generalised into many variants, to name a few, differential-linear cryptanalysis [135], zero-correlation linear cryptanalysis [45], multidimensional linear cryptanalysis [106].

### 2.2.4 Rotational Cryptanalysis

Apart from the above two general methods, rotational cryptanalysis, a dedicated cryptanalysis for analysing ARX constructions, drew a lot of attention since its publication. Although the idea of tracking rotational input pairs can be found before being formalised, see for example, [79, 183], rotational cryptanalysis was first proposed as a generic attack technique against ARX-based designs by Khovratovich and Nikolić in 2010 [117] and applied to Threefish, the ARX-based block cipher underlying the hash function Skein [85]. Most notably, the rotational rebound attack [119], which is an extension of rotational cryptanalysis, is by far the best attack against Skein, to date.

Similar to differential cryptanalysis, rotational cryptanalysis takes advantage of the high probability in the propagation of rotational pairs, *e.g.* $(x, x \lll \gamma)$, through the ARX operations. The following proposition provides a general method to compute the propagation of a rotational pair through modular addition.

**Proposition 1** ([67]). *For independently random variables $x, y \in \mathbb{F}_{2^n}$, and $0 < \gamma < n$,*

$$\Pr[(x \boxplus y) \lll \gamma = (x \lll \gamma) \boxplus (y \lll \gamma)] = (1 + 2^{\gamma-n} + 2^{-\gamma} + 2^{-n})/4\,.$$

**Remark 1.** *The probability is maximized to $2^{-1.415}$ when $n$ is large and $\gamma = 1$. In this chapter, we denote $x \lll 1$ by $\overleftarrow{x}$ to simplify the notation.*

In the original application of rotational cryptanalysis [119], the probability of a rotational distinguisher is determined by the number of modular additions in an ARX system, by assuming the independence in the inputs and outputs of each addition. However, as was shown in [118] that the formula is wrong when applied to Skein and Blake. More specifically, Khovratovich *et al.* pointed out that the

independence assumption in [117] does not hold when an output of a modular addition is directly passed as input into another modular addition. They refer to this event as a "chained modular addition", and show that when such a chain exists, the transition probability over both additions is not independent. Fortunately, the revised result does not invalidate the rotational rebound attack since the probability there was estimated experimentally. A similar effect of dependency was noticed as well for linear cryptanalysis [168], and for differential cryptanalysis [127].

Similar to the modular chains, another issue that was not rigorously analyzed in [117] and its subsequent works, is the injection of constants. The impact of constants to ARX systems is noticed, for example, in [183] where the designers of the block cipher SEA assert that their construction can resist rotational cryptanalysis due to the nonlinear key schedule and the injection of pseudo-random constants. When an ARX structure includes the injection of constants, it is called ARX-C. It was proven in [117] that this structure is complete, *i.e.*, that any function can be implemented through an ARX-C construction. In most papers on rotational cryptanalysis, heuristic experiments are made to estimate the influence of constants.

In the following section, we will show that constants can be beneficial in constructing new distinguishers.

## 2.3 Rotational-XOR Cryptanalysis

In this section, we propose a new cryptanalytic technique called Rotational-XOR cryptanalysis, which is an extension of rotational cryptanalysis by taking the constants in the ARX systems into consideration. We will first define the rotational-XOR differences and subsequently study the theoretical propagation through ARX operations. Afterwards, an example is provided using SPECK to show how the analysis works in real ciphers.

### 2.3.1 Rotational-XOR Difference

In most cases, the constants are injected into the state either through an XOR operation or through modular addition. When the constant $c$ is rotational-invariant, *i.e.*, $c = c \lll \gamma$, for some $\gamma$, XORing with $c$ does not change the rotational property of a rotational pair $(x, x \lll \gamma)$. However, whenever $c$ is not rotational-invariant, the properties of the output require further inspection.

In general, when a constant $c$ that is not rotational-invariant is XORed into a rotational pair $(x, x \lll \gamma)$, the output pair $(x \oplus c, (x \lll \gamma) \oplus c)$ no longer forms a rotational pair. If this pair is given as an input to the modular addition, the basic formula in Proposition 1 for computing the propagation of the rotational property can no longer be used.

**Definition 9** ([7]). *Let $a_1, a_2, x$ be elements in $\mathbb{F}_{2^n}$. We call*

$$(x \oplus a_1, (x \lll \gamma) \oplus a_2)$$

*a pair with $((a_1, a_2), \gamma)$-*rotational-XOR-difference *(or in shorthand notation $((a_1, a_2), \gamma)$-RX-difference and RX-difference when $a_1, a_2, \gamma$ are clear in the context), and such a pair is referred to as an RX-pair.*

It is necessary to point out that in Definition 9, the difference between the values in an RX-pair is not explicitly defined. The reason is that there exist multiple parameters $a_1, a_2$ corresponding to one pair. For instance, by naming a variable $y = x \oplus a_1$, we have $(y, (y \lll \gamma) \oplus ((a_1 \lll \gamma) \oplus a_2))$, which results in the same RX-difference with a different pair. In order to simplify the notation, we give an equivalent definition of an RX-difference below.

**Definition 10** ([146]). *Given a pair of variables $(v_1, v_2) \in \mathbb{F}_{2^n}$, an RX-difference with rotational offset $\gamma$ is defined as*

$$\Delta_\gamma(v_1, v_2) = v_1 \oplus (v_2 \lll \gamma).$$

Note that when $a_1 = a_2 = 0$ or $\Delta_\gamma = 0$, the pair $(v_1, v_2)$ simply becomes a rotational pair. Similar to differential cryptanalysis, an attacker cares less about the exact values of the pair and focuses on the RX-difference. It is possible to take another equivalent way to define RX-differences, while the statistical property revealed is the same.

### 2.3.2 Theoretical Propagation of RX-difference

In this section, we study the propagation of RX-differences through the operations in ARX.

#### The linear operations

Similar to a rotational pair, an RX-pair passes through the linear operations with probability 1. To be specific, rotating a pair $(x \oplus a_1, (x \lll \gamma) \oplus a_2)$ to the left by an offset $\gamma'$ leads to

$$((x \lll \gamma') \oplus (a_1 \lll \gamma'), (x \lll (\gamma' + \gamma) \oplus (a_2 \lll \gamma')).$$

Meanwhile, for XOR we have

$$(x \oplus a_1, (x \lll \gamma) \oplus a_2) \oplus (y \oplus a_1', (y \lll \gamma) \oplus a_2') \\ = ((x \oplus y) \oplus a_1 \oplus a_1', ((x \oplus y) \lll \gamma) \oplus (a_2 \oplus a_2')).$$

With the RX-difference in Definition 10, it means that the linear operations can be directly applied to the RX-difference to deduce the output difference.

### The nonlinear operation

Both rotational relations and differences have a probabilistic propagation through the nonlinear operation $\boxplus$, while the propagation of an RX-difference is more than their simple combination. Here, our goal is to estimate the transition probability with respect to modular addition of two input RX-differences to an output RX-difference. Without loss of generality, we consider the case where the input rotational pairs are $(x \oplus a_1, y \oplus b_1)$ and $(\overleftarrow{x} \oplus a_2, \overleftarrow{y} \oplus b_2)$, and compute the probability of

$$\overleftarrow{(x \oplus a_1) \boxplus (y \oplus b_1) \oplus \Delta_1} = (\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2) \oplus \Delta_2.$$

Firstly, we introduce two lemmata on differential equations of addition, which facilitates our deduction of the main theorem in this section. In the sequel, the identity operation is denoted by Id and left shift by 1 is denoted by SHL.

**Lemma 2** ([175]). *Let $\zeta_1, \zeta_2, \zeta_3 \in \mathbb{F}_{2^n}$ be constants. Let $x, y \in \mathbb{F}_{2^n}$ be independent random variables. The probability of the differential equation*

$$x \boxplus y = (x \oplus \zeta_1) \boxplus (y \oplus \zeta_2) \oplus \zeta_3 \tag{2.1}$$

*is*

$$1_{(\mathrm{Id} \oplus \mathrm{SHL})(\zeta_1 \oplus \zeta_2 \oplus \zeta_3) \preceq \mathrm{SHL}((\zeta_1 \oplus \zeta_3) | (\zeta_2 \oplus \zeta_3))} \cdot 2^{-|\mathrm{SHL}((\zeta_1 \oplus \zeta_3) | (\zeta_2 \oplus \zeta_3))|}, \tag{2.2}$$

*where $a \preceq b$ if $a_i \leq b_i$ for all $i$.*

The following example is provided for a better understanding of Lemma 2.

**Example 1.** *Let $n = 8$, $\zeta_1 = \mathtt{E}, \zeta_2 = \mathtt{9}$ and $\zeta_3 = \mathtt{F7}$ , we have*

$$(\mathrm{Id} \oplus \mathrm{SHL})(\zeta_1 \oplus \zeta_2 \oplus \zeta_3) = \mathtt{10},$$

$$\mathrm{SHL}((\zeta_1 \oplus \zeta_3) | (\zeta_2 \oplus \zeta_3)) = \mathtt{FE},$$

*and*

$$|\operatorname{SHL}((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))| = |\mathtt{FE}| = 7\,.$$

*We evaluate the characteristic function* $1_{(\mathrm{Id} \oplus \operatorname{SHL})(\zeta_1 \oplus \zeta_2 \oplus \zeta_3) \preccurlyeq \operatorname{SHL}((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))}$*, and see that it is equal to 1 since no bit in* $(\mathrm{Id} \oplus \operatorname{SHL})(\zeta_1 \oplus \zeta_2 \oplus \zeta_3)$ *is larger than the respective bit in* $\operatorname{SHL}((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))$*. The probability is then* $2^{-|\operatorname{SHL}((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))|} = 2^{-7}$*.*

**Lemma 3.** *Let* $\zeta_1, \zeta_2, \zeta_3 \in \mathbb{F}_{2^n}$ *be constants. For independent random variables* $x, y \in \mathbb{F}_{2^n}$*, the probability of*

$$x \boxplus y \boxplus 1 = (x \oplus \zeta_1) \boxplus (y \oplus \zeta_2) \oplus \zeta_3 \tag{2.3}$$

*is*

$$1_{(\mathrm{Id} \oplus \operatorname{SHL})(\zeta_1 \oplus \zeta_2 \oplus \zeta_3) \oplus 1 \preceq \operatorname{SHL}((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))} \cdot 2^{-|\operatorname{SHL}((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))|}\,. \tag{2.4}$$

*Proof.* Let $c$ be the carry vector of $x \boxplus y$ (*i.e.*, $c = (x \boxplus y) \oplus x \oplus y$), $z$ the output vector of $x \boxplus y$ (*i.e.*, $z = x \boxplus y$), $h$ the carry vector of $z \boxplus 1$ (*i.e.*, $h = (z \boxplus 1) \oplus z \oplus 1$), and $e = c \oplus 1 \oplus h$. Then,

$$x \boxplus y \boxplus 1 = (x \oplus y \oplus c) \boxplus 1 = x \oplus y \oplus c \oplus 1 \oplus h = x \oplus y \oplus e.$$

Denote the $i$-th bit of the binary expansion of 1 by $1_i$. Then the $i$-th bit of $h$, $h_i$ can be computed recursively through

$$h_i = \begin{cases} 0 & i = 0\,, \\ z_{i-1} \wedge h_{i-1} \oplus z_{i-1} \wedge 1_{i-1} \oplus h_{i-1} \wedge 1_{i-1} & i > 0\,. \end{cases}$$

We get that,

$$h_i = \begin{cases} 0 & i = 0\,, \\ z_0 \wedge h_0 \oplus z_0 \wedge 1_0 \oplus h_0 \wedge 1_0 = x_0 \oplus y_0 & i = 1\,, \\ (x_{i-1} \oplus y_{i-1} \oplus c_{i-1}) \wedge h_{i-1} & i > 1\,. \end{cases}$$

Hence $e_0 = c_0 \oplus 1 \oplus h_0 = 1$ and

$$\begin{aligned} e_{i+1} &= c_{i+1} \oplus h_{i+1} \\ &= x_i \wedge y_i \oplus x_i \wedge c_i \oplus y_i \wedge c_i \oplus (x_i \oplus y_i \oplus c_i) \wedge h_i \\ &= x_i \wedge y_i \oplus x_i \wedge (c_i \oplus h_i) \oplus y_i \wedge (c_i \oplus h_i) \oplus c_i \wedge h_i \\ &= x_i \wedge y_i \oplus x_i \wedge e_i \oplus y_i \wedge e_i \oplus (x_{i-1} \wedge y_{i-1} \oplus x_{i-1} \\ &\qquad \wedge c_{i-1} \oplus y_{i-1} \wedge c_{i-1}) \wedge (x_{i-1} \oplus y_{i-1} \oplus c_{i-1}) \wedge h_{i-1} \\ &= x_i \wedge y_i \oplus x_i \wedge e_i \oplus y_i \wedge e_i \oplus x_{i-1} \wedge y_{i-1} \wedge c_{i-1} \wedge h_{i-1} \\ &= x_i \wedge y_i \oplus x_i \wedge e_i \oplus y_i \wedge e_i\,. \end{aligned} \tag{2.5}$$

The last equation holds since $c_0 \wedge h_0 = 0$ and therefore $c_i \wedge h_i = 0$ for all $i$. In other words, when computing $x \boxplus y \boxplus 1$ we can calculate the carry bit entering position $i+1$ as a function of $x_i, y_i$, and the previous carry bit, $e_i$ by means of Equation (2.5). Notice that the recursive formulae for computing $c = (x \boxplus y) \oplus x \oplus y$ and $e = (x \boxplus y \boxplus 1) \oplus x \oplus y$ are similar except that they start with different initial values; we will use a T-function to derive the probability of the differential equation which is analogous to the XOR-differential probability of addition [160].

We define a T-function for the differential equation $x \boxplus y \boxplus 1 = (x \oplus \zeta_1) \boxplus (y \oplus \zeta_2) \oplus \zeta_3$ as follows.

$$\begin{aligned}
(t_1)_i &= x_i \oplus y_i \oplus e_i \,, \\
e_{i+1} &= x_i \wedge y_i \oplus x_i \wedge e_i \oplus y_i \wedge e_i \,, \\
(t_2)_i &= (x_i \oplus (\zeta_1)_i) \oplus (y_i \oplus (\zeta_2)_i) \oplus g_i \,, \\
g_{i+1} &= (x_i \oplus (\zeta_1)_i) \wedge (y_i \oplus (\zeta_2)_i) \oplus (x_i \oplus (\zeta_1)_i) \wedge g_i \oplus (y_i \oplus (\zeta_2)_i) \wedge g_i \,,
\end{aligned} \tag{2.6}$$

where $e_0 = 1, g_0 = 0$. Let $S_i = (e_i, g_i)$, the T-function is defined as

$$((t_1)_i \oplus (t_2)_i, S_{i+1}) = f(x_i, y_i, (\zeta_1)_i, (\zeta_2)_i, S_i), 0 \le i \le n-1 \,, \tag{2.7}$$

where $f$ follows from Equation (2.6). Define $\omega_i = (\zeta_1)_i || (\zeta_2)_i || (\zeta_3)_i, 0 \le i \le n-1$, $L = (1,1), C = (0,1)^T$ and eight $2 \times 2$ matrices as

$$A_{000} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad A_{001} = A_{010} = A_{100} = \frac{1}{2} \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix},$$

$$A_{111} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad A_{011} = A_{101} = A_{110} = \frac{1}{2} \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}.$$

Using the directed acyclic graph method [160], the probability of Equation (2.3) can be determined by $LA_{\omega_{n-1}} \cdots A_{\omega_1} A_{\omega_0} C$. From [160] we know that the probability of the differential equation $x \boxplus y = (x \oplus \zeta_1) \boxplus (y \oplus \zeta_2) \oplus \zeta_3$ is $LA_{\omega_{n-1}} \cdots A_{\omega_1} A_{\omega_0} C'$, where $C' = (1,0)^T$. The probabilities of Equation (2.1) and Equation (2.3) can be fully determined by the same automaton as shown in Figure 2.2. It starts at state $(1,1)$, reads $\omega_i = (\zeta_1)_i || (\zeta_2)_i || (\zeta_3)_i$ from $i = n-1$ to 0 by regarding $\omega_i$ as the binary representation of a decimal number and updates the automaton. The probability of Equation (2.1) (Equation (2.3) resp.) is nonzero if the automaton ends at state $(1,0)$ (state $(0,1)$ resp.), and every time it passes through a full line, the probability is halved.

We first discuss the condition for the probability to be nonzero, so we omit the value $1/2$ which may occur in the multiplications for now. Since the product

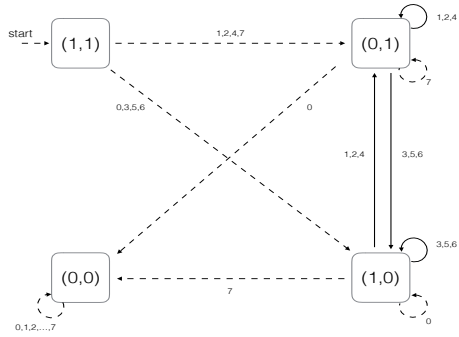



Figure 2.2: Automaton representation of the probability of differential equations.

of $L$ with any matrix $A_j$ can only be $(1,0),(0,1)$ or $(0,0)$, the probability of Equation (2.1) (Equation (2.3) resp.) is nonzero when $LA_{\omega_{n-1}} \cdots A_{\omega_1} A_{\omega_0}$ equals $(1,0)$ ($(0,1)$ resp.). Therefore the probability of Equation (2.3) will be nonzero when the first $n-2$ multiplications do not lead to $(0,0)$, meanwhile $LA_{\omega_{n-1}} \cdots A_{\omega_1} = (0,1)$ and $w_0 = 1,2,4,7$, or $LA_{\omega_{n-1}} \cdots A_{\omega_1} = (1,0)$ and $w_0 = 1,2,4$. Comparing with the probability of Equation (2.1) in the previous lemma, the probability of Equation (2.3) is nonzero if the following condition is satisfied

$$(\mathrm{Id} \oplus \mathrm{SHL})(\zeta_1 \oplus \zeta_2 \oplus \zeta_3) \oplus 1 \preceq \mathrm{SHL}((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3)).$$

If the probability is nonzero, it is determined by the number of times that $\omega_i \in \{001, 010, 100, 011, 101, 110\}$, and each time it contributes $1/2$ to the overall probability except the first multiplication $LA_{\omega_{n-1}}$. Therefore the probability is $2^{-|\,\mathrm{SHL}((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))|}$ since it encounters such matrices when $(((\zeta_1)_i \oplus (\zeta_3)_i)|((\zeta_2)_i \oplus (\zeta_3)_i)) = 1$ for given $i, 0 \le i \le n-1$. Therefore the probability of Equation (2.3) is given by

$$1_{(\mathrm{Id} \oplus \mathrm{SHL})(\zeta_1 \oplus \zeta_2 \oplus \zeta_3) \oplus 1 \preceq \mathrm{SHL}((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))} \cdot 2^{-|\,\mathrm{SHL}((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))|}.$$

which leads to the conclusion. □

Now we are ready to introduce the theorem for deducing the probability in RX-difference propagation through modular addition.

**Theorem 1.** *Let $x, y \in \mathbb{F}_{2^n}$ be independent random variables. Let $a_1, b_1, a_2, b_2, \Delta_1, \Delta_2$ be constants in $\mathbb{F}_{2^n}$. Then,*

$$\begin{aligned}
&\Pr[\overleftarrow{(x \oplus a_1) \boxplus (y \oplus b_1) \oplus \Delta_1} = (\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2) \oplus \Delta_2] \\
&= 1_{(\mathrm{Id} \oplus \mathrm{SHL})(\delta_1 \oplus \delta_2 \oplus \delta_3) \oplus 1 \preceq \mathrm{SHL}((\delta_1 \oplus \delta_3)|(\delta_2 \oplus \delta_3))} \cdot 2^{-|\,\mathrm{SHL}((\delta_1 \oplus \delta_3)|(\delta_2 \oplus \delta_3))|} \cdot 2^{-3} \\
&+ 1_{(\mathrm{Id} \oplus \mathrm{SHL})(\delta_1 \oplus \delta_2 \oplus \delta_3) \preceq \mathrm{SHL}((\delta_1 \oplus \delta_3)|(\delta_2 \oplus \delta_3))} \cdot 2^{-|\,\mathrm{SHL}((\delta_1 \oplus \delta_3)|(\delta_2 \oplus \delta_3))|} \cdot 2^{-1.415},
\end{aligned} \tag{2.8}$$

*where*

$$\begin{aligned}
\delta_1 &= R(a_1) \oplus L'(a_2), \\
\delta_2 &= R(b_1) \oplus L'(b_2),
\end{aligned}$$

*and*

$$\delta_3 = R(\Delta_1) \oplus L'(\Delta_2)\,.$$

*Proof.* Let $C^1$ be the carry vector of $(x \oplus a_1) \boxplus (y \oplus b_1)$ and let $C^1_{n-\gamma}$ be the carry bit in position $n - \gamma$ (*i.e.*, $C^1_{n-\gamma}$ is the most significant carry produced by $(R(x) \oplus R(a_1)) \boxplus (R(y) \oplus R(b_1)))$. We write $\overleftarrow{(x \oplus a_1) \boxplus (y \oplus b_1) \oplus \Delta_1}$ from Equation (2.8) as the concatenation of its left and right parts.

$$\begin{aligned}
&\overleftarrow{(x \oplus a_1) \boxplus (y \oplus b_1) \oplus \Delta_1} \\
&= \overleftarrow{((L(x) \oplus L(a_1)) \boxplus (L(y) \oplus L(b_1)) \boxplus C^1_{n-\gamma}) \oplus L(\Delta_1)||} \\
&\qquad\qquad \overleftarrow{((R(x) \oplus R(a_1)) \boxplus (R(y) \oplus R(b_1))) \oplus R(\Delta_1)} \\
&= ((R(x) \oplus R(a_1)) \boxplus (R(y) \oplus R(b_1))) \oplus R(\Delta_1)|| \\
&\qquad\qquad ((L(x) \oplus L(a_1)) \boxplus (L(y) \oplus L(b_1)) \boxplus C^1_{n-\gamma}) \oplus L(\Delta_1)\,.
\end{aligned}$$

Similarly, let $C^2$ be the carry vector of $(\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2)$, and $C^2_\gamma$ the carry bit in position $\gamma$ (*i.e.*, $C^2_\gamma$ is the most significant carry produced by $((L(x) \oplus R'(a_2)) \boxplus (L(y) \oplus R'(b_2))$. We can write $(\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2) \oplus \Delta_2$ from Equation (2.8) as the concatenation of its left and right parts.

$$
\begin{aligned}
&(\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2) \oplus \Delta_2 \\
&= (\overleftarrow{(L(x)||R(x))} \oplus a_2) \boxplus (\overleftarrow{(L(y)||R(y))} \oplus b_2) \oplus \Delta_2 \\
&= ((R(x)||L(x)) \oplus (L'(a_2)||R'(a_2))) \boxplus ((R(y)||L(y)) \oplus (L'(b_2)||R'(b_2))) \oplus \Delta_2 \\
&= ((R(x) \oplus L'(a_2)) \boxplus (R(y) \oplus L'(b_2)) \boxplus C_\gamma^2) \oplus L'(\Delta_2)|| \\
&\qquad\qquad ((L(x) \oplus R'(a_2)) \boxplus (L(y) \oplus R'(b_2))) \oplus R'(\Delta_2)\,.
\end{aligned}
$$

We get that

$$
\overleftarrow{(x \oplus a_1) \boxplus (y \oplus b_1) \oplus \Delta_1} = (\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2) \oplus \Delta_2
$$

if and only if

$$
\begin{aligned}
&((R(x) \oplus L'(a_2)) \boxplus (R(y) \oplus L'(b_2)) \boxplus C_\gamma^2) \oplus L'(\Delta_2) = \\
&(R(x) \oplus R(a_1)) \boxplus (R(y) \oplus R(b_1)) \oplus R(\Delta_1)\,,
\end{aligned}
\tag{2.9}
$$

and

$$
\begin{aligned}
&((L(x) \oplus L(a_1)) \boxplus (L(y) \oplus L(b_1)) \boxplus C_{n-\gamma}^1) \oplus L(\Delta_1) = \\
&((L(x) \oplus R'(a_2)) \boxplus (L(y) \oplus R'(b_2))) \oplus R'(\Delta_2).
\end{aligned}
\tag{2.10}
$$

Replacing

$$
R(x^\dagger) = R(x) \oplus L'(a_2)
$$

$$
R(y^\dagger) = R(y) \oplus L'(b_2),
$$

we can rewrite the condition in Equation (2.9) as

$$
\begin{aligned}
&R(x^\dagger) \boxplus R(y^\dagger) \boxplus C_\gamma^2 = \\
&(R(x^\dagger) \oplus R(a_1) \oplus L'(a_2)) \boxplus (R(y^\dagger) \oplus R(b_1) \oplus L'(b_2)) \oplus R(\Delta_1) \oplus L'(\Delta_2).
\end{aligned}
\tag{2.11}
$$

Similarly, by setting

$$
L(x^*) = L(x) \oplus L(a_1)
$$

$$
L(y^*) = L(y) \oplus L(b_1),
$$

Equation (2.10) reduces to

$$L(x^*) \boxplus L(y^*) \boxplus C^1_{n-\gamma} =$$

$$((L(x^*) \oplus L(a_1) \oplus R'(a_2)) \boxplus (L(y^*) \oplus L(b_1) \oplus R'(b_2))) \oplus R'(\Delta_2) \oplus L(\Delta_1). \tag{2.12}$$

We can compute the probability that the conditions in Equation (2.11) and Equation (2.12) hold, by means of Lemma 2 and Lemma 3 based on the values of $C^1_{n-\gamma}$ and $C^2_\gamma$.

**Case 1:** $C^2_\gamma = 0$, the probability is the difference propagation probability and it can be calculated by means of Lemma 2.

**Case 2:** $C^2_\gamma = 1$, we solve the differential equations using Lemma 3.

Similarly,

**Case 3:** $C^1_{n-\gamma} = 0$, the probability is the difference propagation probability and it can be calculated by Lemma 2.

**Case 4:** $C^1_{n-\gamma} = 1$, we solve the differential equations using Lemma 3.

When $\gamma = 1$, $L(\cdot), R'(\cdot)$ represent a single bit, hence,

$$C^1_{n-\gamma} = L(a_1) \oplus L(b_1) \oplus L(\Delta_1) \oplus R'(a_2) \oplus R'(b_2) \oplus R'(\Delta_2).$$

In addition, notice that the carry bit of $L(x) \boxplus L(y)$ is independent of that of $R(x) \boxplus R(y)$ when $x, y$ are independent random variables. We have for large $n$ and $\gamma = 1$, $\Pr[C^2_\gamma = 0] = 3/4$ and $\Pr[C^1_{n-\gamma} = 0] = 1/2$, since the carry for 1-bit addition is biased. However, for the addition of two random bit strings, the most significant carry bit can be regarded as balanced. Then,

$$\Pr[C^2_\gamma = 0, C^1_{n-\gamma} = 0] = 2^{-1.415}$$

$$\Pr[C^2_\gamma = 0, C^1_{n-\gamma} = 1] = 2^{-1.415}$$

$$\Pr[C^2_\gamma = 1, C^1_{n-\gamma} = 0] = 2^{-3}$$

$$\Pr[C^2_\gamma = 1, C^1_{n-\gamma} = 1] = 2^{-3}.$$

Therefore, the probability is calculated as

$$\Pr[C^2_\gamma = 0, C^1_{n-\gamma}] \cdot \Pr[x \boxplus y = (x \oplus \delta_1) \boxplus (y \oplus \delta_2) \oplus \delta_3] +$$

$$\Pr[C^2_\gamma = 1, C^1_{n-\gamma}] \cdot \Pr[x \boxplus y \boxplus 1 = (x \oplus \delta_1) \boxplus (y \oplus \delta_2) \oplus \delta_3],$$

which concludes the proof. □

Note that when all the constants are 0, *i.e.*, $a_1 = a_2 = b_1 = b_2 = \Delta_1 = \Delta_2 = 0$, Theorem 1 predicts $\Pr[\overleftarrow{x \boxplus y} = \overleftarrow{x} \boxplus \overleftarrow{y}]$, which is the trivial case.

To be complete, we give an equivalent interpretation of Theorem 1 under Definition 10.

**Corollary 1.** *Suppose that $x, y \in \mathbb{F}_{2^n}$ are independent uniform random variables, $z = x \boxplus y$. For RX-differences $d_x, d_y, d_z \in \mathbb{F}_{2^n}$, and variables $x' = (x \oplus d_x) \ggg 1, y' = (y \oplus d_y) \ggg 1, z' = (z \oplus d_z) \ggg 1$, we have that,*

$$\begin{aligned}\Pr[x' \boxplus y' = z'] = \\ 1_{(\mathrm{Id} \oplus \mathrm{SHL})(\delta_x \oplus \delta_y \oplus \delta_z) \oplus 1 \preceq \mathrm{SHL}((\delta_x \oplus \delta_z) | (\delta_y \oplus \delta_z))} \cdot 2^{-|\mathrm{SHL}((\delta_x \oplus \delta_z) | (\delta_y \oplus \delta_z))|} \cdot 2^{-3} \\ + 1_{(\mathrm{Id} \oplus \mathrm{SHL})(\delta_x \oplus \delta_y \oplus \delta_z) \preceq \mathrm{SHL}((\delta_x \oplus \delta_z) | (\delta_y \oplus \delta_z))} \cdot 2^{-|\mathrm{SHL}((\delta_x \oplus \delta_z) | (\delta_y \oplus \delta_z))|} \cdot 2^{-1.415},\end{aligned} \tag{2.13}$$

*where $\delta_x = L(d_x), \delta_y = L(d_y), \delta_z = L(d_z)$.*

### 2.3.3 An Example in SPECK32/64

Next we give an example of applying RX-cryptanalysis on SPECK32/64 with Theorem 1 when $\gamma = 1$. A detailed rotational-XOR cryptanalysis with automatic search techniques on other instances of SPECK will be discussed in Section 2.5.2.

Using a greedy algorithm, we obtained a 6-round characteristic with RX-differences for the key-schedule of SPECK32/64. In Table 2.1 we compare the probability predicted by Theorem 1 and the probability obtained by iterating all $2^{32}$ possible $(x, y)$ with a fixed tuple $(a_1, b_1, \Delta_1, a_2, b_2, \Delta_2)$. As is evident from Table 2.1, the values match perfectly. In addition, the empirical probability of the characteristic over $2^{33}$ uniformly chosen keys is around $2^{-25.046}$. Interestingly, the empirical probability of the full characteristic is lower than the one predicted in Table 2.1, which means the left and right inputs to the round function are not independent. Nevertheless, this characteristic suggests that a weak-key class of size $2^{-25} \cdot 2^{64} = 2^{39}$ exists for 7-round SPECK32/64.

The characteristic in the key schedule can be utilised to construct a 7-round RX-distinguisher for SPECK32/64. We started by generating a 64-bit random master-key and checked if it belongs to the weak-key class (*i.e.*, if the resulting subkeys satisfy the characteristic in Table 2.1). Once an appropriate key was found, we used it to encrypt $2^{32}$ chosen plaintexts with a $((0, 0), 1)$-RX difference.

Table 2.1: A table comparing the transition probability predicted through Theorem 1 and the empirical probability for uniformly chosen $x$ and $y$, and a fixed $(a_1, b_1, \Delta_1, a_2, b_2, \Delta_2)$. The rotational amount is $\gamma = 1$. All RX-differences are in hexadecimal notation.

| Round | $a_1$ | $b_1$ | $\Delta_1$ | $a_2$ | $b_2$ | $\Delta_2$ | Predicted Prob. | Empirical Prob. |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | $2^{-1.415}$ | $2^{-1.415}$ |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | $2^{-1.415}$ | $2^{-1.415}$ |
| 3 | 0 | 1 | 0 | 0 | 1 | 2 | $2^{-2.415}$ | $2^{-2.415}$ |
| 4 | 0 | 2 | 6 | 0 | 0 | 8 | $2^{-2.415}$ | $2^{-2.415}$ |
| 5 | 0 | D | C4 | 0 | B | 78 | $2^{-6.415}$ | $2^{-6.415}$ |
| 6 | 0 | F4 | 0 | 1000 | 50 | 1088 | $2^{-7.415}$ | $2^{-7.415}$ |
| Total | | | | | | | $2^{-21.49}$ | |

Using Theorem 1, we found a possible characteristic taking into account the RX-difference propagation through the modular addition, and the RX-difference coming through the key injection.

We repeated this experiment using $2^7$ weak keys. The average number of keys to discard before finding a key following the characteristic in the key schedule was approximately $2^{25.1}$, so the weak-key class is indeed of size $2^{39}$. In Table 2.2 we present the characteristic, the predicted probability, and the average empirical probability. The average number of input pairs with a ((0,0),1)-difference following the full 7-round characteristic is 1.33, hence the probability is $2^{-31.58}$. Moreover, when taking the differential effect into account (*i.e.*, only checking how many pairs satisfy the required RX-difference in the last round), the average number of such pairs is 3.83, thus the probability is $2^{-30.06}$.

## 2.4 A General Framework for Automatic Search on ARX Primitives

For SPN ciphers, designers often follow the Wide Trail Strategy to support the argument for a provable resistance against differential and linear cryptanalysis. In order to set upper bounds to the maximum differential probability of the differential characteristics and the maximum linear correlation of the linear trails, we often choose:

- the S-boxes to be such that the difference propagations have a low

Table 2.2: A table describing the RX-distinguisher for 7-round SPECK32/64. All RX-differences are in hexadecimal notation, and $\gamma = 1$.

| Round | Difference (left,right) | Key diff. | Predicted accum. prob. | Empirical accum. prob. |
|---|---|---|---|---|
| 0 | 0, 0 | 0 | | |
| 1 | 0, 0 | 0 | $2^{-1.415}$ | $2^{-1.415}$ |
| 2 | 0, 0 | 3 | $2^{-2.83}$ | $2^{-2.85}$ |
| 3 | 3, 3 | 4 | $2^{-4.245}$ | $2^{-4.27}$ |
| 4 | 607, 60B | 11 | $2^{-8.66}$ | $2^{-8.68}$ |
| 5 | 40E, 1C22 | 1B8 | $2^{-15.075}$ | $2^{-15.01}$ |
| 6 | 3992, 491A | 1668 | $2^{-21.49}$ | $2^{-21.44}$ |
| 7 | 333F, 1756 | | $2^{-31.905}$ | $2^{-31.6}$ |

probability and the linear approximations also have a low absolute correlation;

- the linear layer to be such that the minimum number of active S-boxes is relatively high.

For instance, the S-box of the AES is derived from the inverse function over $\mathbb{F}_{2^8}$, which is differentially 4-uniform. The linear layer is the composition of ShiftRows which achieves good diffusion property, and MixColumns which provides extra diffusion by an MDS matrix. With such a combination, the probability of 4-round differential characteristics in the AES can be upper bounded.

However, for ciphers without S-boxes, especially, ARX ciphers, there is no formula to deduce the security margin, not to mention a simple way to find differential characteristics and linear trails. Previously, finding such characteristics requires a large amount of human effort and experience, an astonishing example is the cryptanalysis of the hash function SHA-1 and MD5 [202, 203], where the distinguishers were found manually. In this section, we will introduce a generic model for automatic search, aided by software search engines; then linear cryptanalysis and RX-cryptanalysis are applied to an ARX cipher as a demonstration of our toolbox for cryptanalysis.

### 2.4.1 A General Model for Automatic Search

As we have recalled in Section 2.2, the hypothesis of stochastic equivalence is the basis for constructing statistical distinguishers. It allows us to focus on some distinguishing properties over each individual round, and then connect them under a certain strategy to optimise the strength of the overall distinguisher.

Here we describe a model for searching for optimal statistical distinguishers in block ciphers. Given a block cipher with $n$-bit block size and $r$ rounds, we track the propagation of values (*e.g.*, differences, masks, RX-differences, etc.) through the round functions of a block cipher with a non-zero probability/correlation, which can be illustrated as a shortest path problem in a weighted directed acyclic graph as follows.[3]

**Find a shortest path in a weighted directed acyclic graph**

- Model an $(r+1)$-partite graph $G = (V, E)$.
- The vertice set $V = \bigcup_{i=0}^{r} V_i$, where $V_i = \mathbb{F}_{2^n}$.
- A directed edge is drawn from $v_i \in V_i$ to $v_{i+i} \in V_{i+1}$ when the propagation of $v_i$ to $v_{i+1}$ through the $i$-th round has a non-zero probability (correlation, resp.). The absolute binary logarithm of the probability (the absolute value of the correlation, resp.) is assigned as the weight of the edge.
- Finding an optimal distinguisher is equivalent to finding a shortest path of length $r$ in $G$.

Note that the search of truncated differentials in an SPN cipher is analogous to the above model, where the vertice set contains all the truncated differences (active/inactive) instead of actual ones.

Although the shortest path problem is well-studied and can be solved with a number of efficient algorithms such as Dijkstra's algorithm, a direct application is infeasible for most cryptographic primitives, due to the tremendous size of the vertice set. For instance, for a 32-bit block cipher with $r$ rounds, the number of vertices is $(r+1) \cdot 2^{32}$, which makes it impossible to efficiently store and access them on a personal computer. Dedicated works have been developed on ad hoc strategies, such as the branch-and-bound method by Matsui [152] in searching linear trails, as well as a variety of optimised tools including YAARX [197],

[3] We refer to the textbook [204] for terminology in graph theory.

ARX Toolkit [140] and algorithms designed by Biryukov *et al.* and Bouillaguet *et al.* [40, 41, 49].

Another commonly used approach in recent years is to utilise some search engine, which avoids customising the search algorithms from scratch and saves cryptanalysts from dealing with complex programs. In the next section, we will study this approach in detail, which is simply referred to as automatic search in this thesis.

### 2.4.2 State-of-the-art Search Engines

#### SAT/SMT

The Boolean satisfiability problem (SAT) considers whether there is a valid assignment to Boolean variables satisfying a given set of Boolean clauses. A Boolean clause consists of Boolean variables (called literals), operators AND, OR, NOT, and parentheses. For example, the clause $x$ AND (NOT $y$) is satisfiable since $x =$ TRUE, $y =$ FALSE is a valid assignment.

The SAT problem is NP-complete [59]. However for most practical situations, the solutions can be found in a reasonable time. There are a large number of heuristic SAT solvers, and all of them accept the DIMACS CNF (Conjunctive Normal Form) files as the standard input format. In the CNF format, all clauses are literals with logical operation OR and NOT, while the clauses are concatenated by AND. The output is either *satisfiable* or *unsatisfiable*. When it is satisfiable, the solver can also return a valid assignment to all literals. More specifically, SAT solvers will start searching with an initial assignment, then calculate the number of conflicting clauses, based on which the search tree of the SAT solver decides the next step of searching to eliminate possible conflicts until a valid or no solution is found.

In general, any Boolean or arithmetic operation can be realised by a combination of ANDs, ORs and NOTs. When considering problems in cryptanalysis, XOR is one of the most common operations. If we translate XOR clauses into CNF, a sentence $a \oplus b$ becomes two clauses $(\neg a \vee \neg b) \wedge (a \vee b)$. In general, the XOR of $n$ Boolean variables will give $2^{n-1}$ clauses in CNF format. Even if the expressions are logically equivalent, the underlying structure of the XOR equation system is missing in terms of the CNF format. A system of XOR equations is in fact a linear equation system over $\mathbb{F}_2$, therefore, it can be solved by Gaussian elimination in time $O(n^3)$, where $n$ is the number of variables. In many circumstances, Gaussian elimination is much more efficient than translating XOR into operations $\vee$ and $\wedge$. One SAT solver called Cryptominisat4 [181, 182]

is specially designed to be compatible with XOR operations and solve the XOR equation system by Gaussian elimination. Another example is inequality. When comparing Boolean expressions, $z \geq a \oplus b$, it is equivalent to *if* $a \oplus b$*, then* $z$, which is logically consistent with $(\neg a \vee b \vee z) \wedge (a \vee \neg b \vee z)$.

Addition over integers is an unnatural operation in the SAT language, as it is not easy to describe with only OR and AND. In SAT theory, constraints like the objective function $\sum_i x_i \leq k$, where $k \geq 1$, are called cardinality constraints, which belongs to an even larger class called Pseudo Boolean constraints (PB-constraints). There are two directions to handle the cardinality constraints: one is to develop new PB-solvers dedicated to cardinality constraints, the other one is to convert cardinality constraints into CNF format. One plain method is to enumerate all the possible combinations of no more than $k$ out of $n$ variables being true , *i.e.* the conjunction of $\binom{n}{k+1}$ clauses $\bigwedge_{i_1,\ldots,i_{k+1}} (\neg x_{i_1} \vee \cdots \vee \neg x_{i_{k+1}})$. However it is not applicable when $n, k$ are large. Throughout the literature, a large number of methods to encode the cardinality constraints are presented. The basic idea is to add new variables to reduce the number of constraints. Since it is a trade-off between the number of new variables needed and the number of clauses, while the sizes of variables and clauses both have a significant influence on the efficiency of solving, it is critical to find a good encoding method. In this section, we use the sequential encoding method [180], as shown in Equation (2.14). For $\sum_i x_i \leq k$, new dummy variables $\{u_{i,j}\}_{1 \leq i \leq n-1, 1 \leq j \leq k}$ are introduced to return contradiction when the cardinality is larger than $k$.

$$\begin{cases} (\neg x_1 \vee u_{1,1}) \wedge (\neg u_{1,j}), \\ (\neg x_i \vee u_{i,1}) \wedge (\neg u_{i-1,1} \vee u_{i,1}) \wedge (\neg x_i \vee \neg u_{i-1,j-1} \vee u_{i,j}) \\ \qquad\qquad \wedge (\neg u_{i-1,j} \vee u_{i,j}) \wedge (\neg x_i \vee \neg u_{i-1,k}), \\ \neg x_n \vee \neg u_{n-1,k}, \end{cases} \tag{2.14}$$

where $1 < j \leq k, 1 < i < n$. The sequential encoding of cardinality constraints is one of the best methods, with a relatively small amount of additional variables and a great reduction of clauses. When $k = 0$, all variables are zero, which can be translated to $n$ clauses as $\neg x_i$, $1 \leq i \leq n$.

As a generalisation, satisfiability modulo theory (SMT) problem supports a larger variety of data structures and functionalities than SAT, such as bit vectors and self-defined functions. Dozens of SMT solvers have been developed in the last decades; widely used examples are STP [89] and Z3 [70], which are SMT solvers based on a SAT solver. STP and Z3 read files in the SMTLIB [15] language, then convert them into a CNF file which can be solved by an underlying SAT solver. The solvers support most operations on Boolean vectors, and integrate arithmetic operations including addition and

multiplication (packaging cardinality constraints with a similar approach as Equation (2.14) into a function), which enables to model complex problems.

### Mixed-Integer Linear Programming

Mixed-Integer Linear Programming (MILP) was adopted in modelling cryptanalysis problems by Borghoff *et al.*, for example, [48], and later improved in a number of studies [161, 189]. Linear programming language naturally contains arithmetic operations such as addition, while the sentences for describing Boolean operations can be complicated. Nevertheless, MILP has been successfully applied to automatic search of many distinguishers [87, 174, 189, 209].

SMT has certain similarity with the 0-1 integer programming problem or MILP in interpreting problems, while the underlying ideas to solve them differ significantly. For the MILP problem, linear programming solvers first regard the problem as a general linear programming problem over the real numbers, then by Branch and Cut, they carefully rule out illegal branches and then limit the solution to 0-1 integers. SMT solvers try to translate the problem to SAT, then solve it within a binary field. Due to the different methodologies of solvers, the performance depends heavily on the background and structure of the underlying problem.

### Constraint Programming

A constraint programming (CP) problem states variables and their relations in constraints. CP solvers have been applied to solving cryptanalytic problems in [92, 188]; they take advantage of the functionality in CP solvers that the constraints over variables can be expressed in look-up tables, such that difference distribution table of 8-bit S-boxes can be efficiently processed. Currently, CP solvers such as Choco [170] support Boolean, integer, real variables, but are less flexible than SMT solvers in handling bitvectors.

The models for different search engines are mutually translatable, while the main difference is on the performance which depends on the specific problem. In this chapter, and the following chapters whenever applicable, we adopt SAT/SMT solvers.

### 2.4.3 Search Strategy

In order to obtain an optimal characteristic, an objective function to minimise the overall weight is necessary. As we have shown in Section 2.4.2, objective functions are supported naturally in MILP, however, not for SAT/SMT which is adopted in our research. Instead, we set cardinality constraints and finely adjust the bound until the minimum is located. In such a case, the strategy of searching, *i.e.*, choosing appropriate bounds, is crucial to the efficiency. To avoid being ambiguous, the cardinality constraints are also referred to as objective functions, due to their identical functionality. In the following, we discuss the design of search strategy, which will be adopted and further optimised in Section 2.5.2.

#### Vanilla Strategy

Obviously, there exist trivial upper and lower bounds for the objective function. As a basic strategy, one may start with the trivial bound and then test one-by-one with a decreasing/increasing order. It is believed that, for cryptographic problems executing on SAT solvers, the time for an *unsatisfiable* decision takes much longer than *satisfiable* one, because the search is roughly brute-force before returning the decision for an *unsatisfiable* [181]. Hence, we follow Algorithm 1 to find a shortest path of length $r$ in graph $G$ with the weight of a shortest path of length $r-1$.

**Algorithm 1** Find the weight of a shortest path in graph $G$

**Input:** A shortest path $p_0$ in $G$ of length $r-1$ with weight $\ell$
**Output:** The weight of a shortest path $p_1$ in $G$ with length $r$

1: Extend the length of $p_0$ by one with a cost as small as possible, pass its weight in a variable $\ell'$
2: Set the objective function to $\ell'$ and run the solver
3: **while** The problem is *satisfiable* with upper bound $\ell'$ **do**
4: $\quad \ell' \leftarrow \ell' - 1$
5: $\quad$ Run the solver
6: **return** $\ell' + 1$

It is ensured that the *unsatisfiable* decision only happens once. However, if a direct extension of the path $p_0$ found by the solver is of relatively high weight, which usually happens in ARX-related problems, it is intolerable due to the exponentially-growing decision time for *satisfiable* when the size of the problem is sufficiently large, let alone the time consumed to find such an extension at

a minimum cost. Therefore, we involve the idea of binary search in the next strategy.

#### Binary-Search Strategy

In order to find a target in a sorted array, instead of going through them one-by-one from the end of the array, binary search starts from the middle and each decision chops off the possible interval by half. In our problem, the interval is $(\ell, U]$, where $\ell$ is the weight of the shortest path of length $r-1$ in $G$ and $U$ is a trivial upper bound (for instance, the block size $n$ since characteristics with probability less than $2^{-n}$ is less interesting). We illustrate the strategy in the following algorithm.

**Algorithm 2** Find the weight of a shortest path in graph $G$

**Input:** A search interval $(\ell, U]$
**Output:** The weight of a shortest path in $G$ with length $r$

Set the search interval to $[T^-, T^+]$
$T^- \leftarrow \ell$,$T^+ \leftarrow \lfloor \frac{\ell+U}{2} \rfloor$
**while** $T^- < T^+$ **do**
    Run the solver
    **if** the problem is *satisfiable* **then**
        $U \leftarrow T^+$
        $T^+ \leftarrow \lfloor \frac{T^-+T^+}{2} \rfloor$
    **else**
        $\ell \leftarrow T^-$
        $T^- \leftarrow T^+$, $T^+ \leftarrow U$
**return** $T^- + 1$

**Remark 2.** *Note that the second strategy may encounter more unsatisfiable decisions than the first, but it can still be better in performance. In practice, the efficiency of the strategy largely depends on the underlying problem, and dedicated improvements can be applied to the above strategies for specific problems.*

### 2.4.4 A Fully Automated Tool ARXPY

As we have already mentioned, automatic search techniques have been applied to many ciphers in terms of various cryptanalysis methods. To the best of our knowledge, previous automated tools [128, 140, 141, 197] searching for statistical characteristics are implemented specifically for a particular cipher or

a family of them. Anyone who attempts to use the tool for a cipher which is not implemented in the package, has to figure out the mechanism of the tool and write the code by oneself, which is inefficient as well as going against the original intention of the authors who share their code online.

Our goal is to push the automation process one step further, where it allows the users to utilise and extend the package to new ciphers by simply implementing the cipher in a way that the tool also recognises. ArxPy is such a tool for finding optimal RX-characteristics in ARX block ciphers. In a nutshell, ArxPy takes a Python-like implementation of an ARX block cipher as the input, translates the problem into SAT/SMT files automatically, and gives it as an input to the solver. Given a Python implementation of an ARX block cipher, ArxPy is executed with a simple shell command. Therefore, the only effort to use ArxPy is implementing the ARX block cipher in Python, comparing with generating the files manually. On top of that, ArxPy is open source and has a modular architecture, therefore, it can be easily adapted for specific scenarios.

ArxPy has been implemented in three modules: the ARX block cipher parser, the SMT writer and the characteristic finder. Here we briefly explain these three modules.

The parser module takes a specific implementation of an ARX cipher in Python. The implementation contains the block size, plaintexts, ciphertexts and round keys as the variables, the key schedule and the round function as the functions. Then, it generates a sequence of symbolic expressions of the output values and the round keys. SymPy [157], a Python library for symbolic mathematics, is used to generate and parse the symbolic expressions.

The writer module takes the symbolic expressions generated by the parser as an input, and outputs an SMT file. This is done by extracting the sequence of ARX operations of the encryption algorithm and the key schedule, and by translating these operations into equations. The sequence of operations is obtained from the symbolic expressions by traversing them as trees and extracting their nodes with SymPy.

Finally, the finder module implements the search strategy to find the optimal RX-characteristics with the STP solver. The binary search strategy described in Section 2.4.3 is used to minimize the weight of the characteristic. To find the optimal RX-characteristic, ArxPy searches for characteristics up to many weights. After the execution finishes, the last characteristic in the output file is the optimal one found by the tool.

## 2.5 Applications to the SPECK Family

In this section, we apply the general framework of automatic search to the block cipher SPECK with respect to linear cryptanalysis in Section 2.5.1 and RX-cryptanalysis in Section 2.5.2. First, we model the graph by studying the representation of the edges and their weights, and then show characteristics found by the solver.

### 2.5.1 Automatic Search in Linear Cryptanalysis

We call a linear trail over a (round-reduced) cipher with maximum correlation amplitude an optimal linear trail. Here, we apply our automatic search framework to searching for optimal linear trails in round-reduced SPECK, which gives a general idea on the resistance of SPECK towards linear cryptanalysis. We first transform the rules for linear masks in ARX operations into SAT/SMT clauses, then apply the model for SPECK to find linear trails and linear hulls. Afterwards, we compare and discuss the runtime of different SAT/SMT solvers.

#### Translating Clauses for Modular Addition

The behaviours of masks through linear operations are easy to describe, since the correlation is either zero or $\pm 1$. For example, with input masks $\Gamma_a, \Gamma_b$ and output mask $\Gamma_c$, the condition for being a linear approximation of XOR with nonzero correlation is $\Gamma_a = \Gamma_b = \Gamma_c$. The condition for being a nontrivial linear approximation of three-fork branching is $\Gamma_a \oplus \Gamma_b \oplus \Gamma_c = 0$, and the conditions for rotational circular shift is equivalent to each corresponding bit of the masks.

However for the nonlinear operation modular addition, it is necessary to have a better understanding on the nature of addition modulo $2^n$. The seminal works on linear correlation of modular addition are by Wallén *et al.* [168, 199]. They propose a recursive method to calculate the correlation of a linear approximation in addition modulo $2^n$ efficiently by an automaton. The only drawback of the recursive automaton is that it is very difficult to translate the expression into bit-level linear relations in masks, *i.e.* every bit is dependent on all previous bits, which leads to a huge number of complex constraints. Therefore, even though there are several papers discussing the heuristic search methods of differential characteristics, no previous result finds linear trails in ARX ciphers with SAT theory.

In order to avoid the recursive expression, an explicit result on calculating the correlation of linear approximations in modular addition is proven by

Schulte-Geers [175]. Despite the recursive property of the carry, modular addition is CCZ-equivalent to a vectorial quadratic Boolean function. A more natural formula to calculate the correlation in addition modulo $2^n$ is given in Proposition 2.

**Proposition 2** ([175])**.** *Let $z$ be an $n$-bit vector satisfying $z \oplus (z \gg 1) \oplus ((u \oplus v \oplus w) \gg 1) = 0$, $z_{n-1} = 0$, where $u$ is the output mask, $v, w$ are the input masks in a linear approximation of addition modulo $2^n$. Then the correlation of the linear approximation is given by*

$$C((u, v), w) = 1_{u \oplus v \preccurlyeq z} 1_{u \oplus w \preccurlyeq z} (-1)^{(u \oplus w) \cdot (u \oplus v)} 2^{-|z|}.$$

Comparing to a recursive algorithm, the Hamming weight of $z$ determines the amplitude of the correlation directly, while each bit of $z$ can be explicitly calculated from input and output masks.

From Proposition 2, to obtain a valid linear approximation, the input masks $v, w$ and output mask $u$ through addition modulo $2^n$ need to follow the constraints below.

$$\begin{aligned} &z_{n-1} = 0, \\ &z_{n-2} = u_{n-1} \oplus v_{n-1} \oplus w_{n-1}, \\ &z_j = z_{j+1} \oplus u_{j+1} \oplus v_{j+1} \oplus w_{j+1}, \\ &z_i \geq u_i \oplus v_i, \\ &z_i \geq u_i \oplus w_i, \end{aligned} \tag{2.15}$$

where $0 \leq i \leq n-1$, $0 \leq j \leq n-3$.

### From Linear Relations Towards SATisfiability

We take Speck32 as an example for simplicity. Figure 2.3 shows the notation of the masks in round $r$. From Equation (2.15), we can derive the constraints for a linear approximation of Speck32 in round $r$ as

$$\begin{aligned} z_{15}^r = 0, \quad z_{14}^r &= \Gamma a_6^r \oplus \Gamma c_{15}^r \oplus \Gamma d_{15}^r, \\ z_j^r &= z_{j+1}^r \oplus \Gamma a_{j+8}^r \oplus \Gamma c_{j+1}^r \oplus \Gamma d_{j+1}^r, \\ z_i^r &\geq \Gamma a_{i+7}^r \oplus \Gamma d_i^r, \quad z_i^r \geq \Gamma c_i^r \oplus \Gamma d_i^r, \\ \Gamma d_i^r = \Gamma a_i^{r+1} \oplus \Gamma b_i^{r+1}, \quad &\Gamma c_i^r = \Gamma b_i^r \oplus \Gamma b_{i+2}^{r+1}, \end{aligned} \tag{2.16}$$

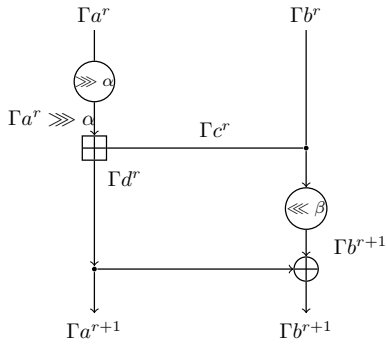



Figure 2.3: Notation of masks in round function of SPECK32.

where $0 \leq i \leq 15$, $0 \leq j \leq 13$, and $\sum_{r,i} z_i^r$ is to be minimized.

Note that when the block size is 32, it is preferable to take the vanilla strategy as shown in Algorithm 3.

**Algorithm 3** Find an optimal linear trail

**Input:** An optimal linear trail $L$ with correlation $2^{-\ell}$ of an $r$ round-reduced cipher

**Output:** The correlation of the optimal linear trail in $r+1$ round-reduced cipher

1: Append a 1-round trail at the end of $L$ to extend it into a $r+1$ round valid linear trail $L'$ with correlation $2^{-\ell'}$
2: **while** the problem is *satisfiable* with $\sum_{r,i} z_i^r \leq \ell'$ **do**
3: $\quad \ell' \leftarrow \ell' - 1$
4: **return** $2^{-(\ell'+1)}$

An overview on the correlation of optimal linear trails in round-reduced SPECK ciphers is given in Table 2.3. Our experiments for searching optimal linear trails were performed on a PC with 8 Intel® Core™ i7 processors clocked at 3.40 GHz. In order to speed up the search by utilising the parallel mode in Cryptominisat4, we run the program on a cruncher with 40 Intel® Xeon™ E5-2687W v3 processors clocked at 3.1 GHz. We confirm all the correlations of optimal linear trails in [211]. Moreover, our method covers significantly more rounds in larger versions of SPECK: 11/13/9/9 rounds comparing to 7/5/4/4 rounds in the previous paper [211] for SPECK48/64/96/128.

We also show examples of linear trails with the best correlation for round-reduced SPECK in Table 2.4. Sometimes without further constraints, input and

Table 2.3: Correlation of the best linear trails in the SPECK family.

| R | SPECK32 | R | SPECK32 | R | SPECK48 | SPECK64 | SPECK96 | SPECK128 |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 12 | $2^{-20}$ | 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 13 | $2^{-22}$ | 2 | 1 | 1 | 1 | 1 |
| 3 | $2^{-1}$ | 14 | $2^{-24}$ | 3 | $2^{-1}$ | $2^{-1}$ | $2^{-1}$ | $2^{-1}$ |
| 4 | $2^{-3}$ | 15 | $2^{-26}$ | 4 | $2^{-3}$ | $2^{-3}$ | $2^{-3}$ | $2^{-3}$ |
| 5 | $2^{-5}$ | 16 | $2^{-28}$ | 5 | $2^{-6}$ | $2^{-6}$ | $2^{-6}$ | $2^{-6}$ |
| 6 | $2^{-7}$ | 17 | $2^{-30}$ | 6 | $2^{-8}$ | $2^{-9}$ | $2^{-9}$ | $2^{-9}$ |
| 7 | $2^{-9}$ | 18 | $2^{-34}$ | 7 | $2^{-12}$ | $2^{-13}$ | $2^{-13}$ | $2^{-13}$ |
| 8 | $2^{-12}$ | 19 | $2^{-36}$ | 8 | $2^{-15}$ | $2^{-17}$ | $2^{-18}$ | $2^{-18}$ |
| 9 | $2^{-14}$ | 20 | $2^{-38}$ | 9 | $2^{-19}$ | $2^{-19}$ | $2^{-22}$ | $2^{-22}$ |
| 10 | $2^{-17}$ | 21 | $2^{-40}$ | 10 | $2^{-22}$ | $2^{-21}$ | | |
| 11 | $2^{-19}$ | 22 | $2^{-42}$ | 11 | $2^{-25}$ | $2^{-24}$ | | |
| | | | | 12 | | $2^{-27}$ | | |
| | | | | 13 | | $2^{-30}$ | | |

output masks may have very high Hamming weight. By setting cardinality constraints on the Hamming weights of the masks, we can obtain trails with input and output masks of the lowest Hamming weight under a given correlation and number of rounds; an example is the linear trail of 11-round SPECK32 in Table 2.4.

### Enumerating Linear Trails in a Linear Hull

For most SAT solvers, if the problem is satisfiable, they can print all the solutions. However, due to the additional variables introduced by encoding methods in generating the CNF files, the solvers may output duplicated solutions which represent the same trail, as also observed by Kölbl *et al.* in [130]. To avoid inaccuracy, we generate the solutions one by one:

**Step 1:** Generate the CNF file for the problem, and ask the solver to give one solution $\bar{s}$ if it exists.

**Step 2:** Append a new clause to the current CNF file in order to rule out $\bar{s}$.

**Step 3:** Ask the solver to give a solution, repeat step 2 until the solver returns *unsatisfiable*.

In Table 2.5, we give the best linear hulls found and their corresponding distribution of trails for 9-round and 10-round SPECK32, where the averaged

Table 2.4: Linear trails with best correlation in reduced-round Speck.

| R | Speck32 | | Speck48 | | Speck64 | |
|---|---|---|---|---|---|---|
| 1 | 4000 | 00B0 | 800121 | 158021 | 00101800 | 00001812 |
| 2 | 0000 | 00C0 | 018100 | 200101 | 00001000 | 00000010 |
| 3 | 0300 | 0300 | 000100 | 000001 | 00000018 | 00000000 |
| 4 | 0C1E | 0818 | 000001 | 000000 | D8000000 | C0000000 |
| 5 | F000 | D010 | 098000 | 080000 | 04100006 | 04800006 |
| 6 | 4683 | 4743 | 406100 | 406800 | 0026D030 | 0420C030 |
| 7 | 00A0 | 0629 | 00024B | 00420A | 01070101 | 21073781 |
| 8 | 78A0 | 18A1 | 001040 | 5E1042 | 01B00100 | 00318601 |
| 9 | 0090 | 6021 | 9082C0 | F082D0 | 01800001 | 0181B000 |
| 10 | 6080 | 4081 | 000018 | 80D09B | 01000000 | 00018000 |
| 11 | 0080 | 0001 | DE84DC | C684DC | 00010000 | 00000000 |
| 12 | 0001 | 0000 | | | 00000D00 | 00000C00 |
| 13 | | | | | 00006065 | 00006068 |

| R | Speck96 | | Speck128 | |
|---|---|---|---|---|
| 1 | 000001800120 | 140000018021 | 0000000001800120 | 1400000000018021 |
| 2 | 000000018100 | 200000000101 | 0000000000018100 | 2000000000000101 |
| 3 | 000000000100 | 000000000001 | 0000000000000100 | 0000000000000001 |
| 4 | 000000000001 | 000000000000 | 0000000000000001 | 0000000000000000 |
| 5 | 098000000000 | 080000000000 | 0D00000000000000 | 0C00000000000000 |
| 6 | 404000000000 | 404800000000 | 6040000000000000 | 604C000000000000 |
| 7 | 000000000002 | 004000000002 | 0000000000000003 | 0060000000000003 |
| 8 | 180000000010 | 1A0000000010 | 1800000000000018 | 1B00000000000018 |
| 9 | 009000000080 | 108000000080 | 00900000000000C0 | 18800000000000C0 |
| 10 | 440458000404 | 840480000404 | 0000000004045E06 | C404800000000606 |

linear probability ALP is estimated by summing up the squared correlation of all linear trails found by the solver. The experimental average ALP with 128 random keys for the above linear hulls are $2^{-28.9}$ and $2^{-31.1}$ respectively.

### Comparison of Solvers

In some previous papers on automatic searching of differential and linear trails, *e.g.* [161, 189], the search problem is modelled as a MILP problem and solved by CPLEX. To compare the performance of CPLEX and Cryptominisat4, we encode the same constraints with the MILP language and CNF without optimisation. Despite the connection between the MILP and the SAT problem

Table 2.5: The distribution of linear trails in the best found 9-/10-round Speck32 linear hull.

| Cor. | 9-round* #trails | Cor. | 10-round† #trails |
|---|---|---|---|
| $2^{-14}$ | 0 | $2^{-17}$ | 1 |
| $2^{-15}$ | 1 | $2^{-18}$ | 1 |
| $2^{-16}$ | 0 | $2^{-19}$ | 6 |
| $2^{-17}$ | 3 | $2^{-20}$ | 16 |
| $2^{-18}$ | 2 | $2^{-21}$ | 81 |
| $2^{-19}$ | 21 | $2^{-22}$ | 344 |
| $2^{-20}$ | 69 | $2^{-23}$ | 1298 |
| $2^{-21}$ | 346 | $2^{-24}$ | 4873 |
| $2^{-22}$ | 1196 | $2^{-25}$ | 17781 |
| $2^{-23}$ | 4461 | $2^{-26}$ | $\geq$60480 |
| $2^{-24}$ | 15241 | $2^{-27}$ | $\geq$23951 |
| $2^{-25}$ | 48397 | $2^{-28}$ | $\geq$11272 |
| $2^{-26}$ | | $2^{-29}$ | $\geq$3789 |
| $2^{-27}$ | | $2^{-30}$ | $\geq$5883 |
| $2^{-28}$ | | $2^{-31}$ | $\geq$48951 |
| ALP | $2^{-29.1}$ | ALP | $\geq 2^{-32.1}$ |

*input masks: `0010`, `1400`, output masks: `0B00`, `0800`
†input masks: `0000`, `0306`, output masks: `0B00`, `0800`

with an objective function, our method has an advantage over CPLEX. For instance, to find an optimal linear trail in 6-round Speck32, it takes over 4000 seconds on CPLEX, compared to about 2 seconds on Cryptominisat4.[4]

Another commonly used solver is STP [89], which is an SMT solver and also a CNF generator. It encodes constraints into a CNF file inside the solver based on the SMTLIB2 language, and then calls a SAT solver to solve the problem. Unlike Cryptominisat4, STP does not support XOR clauses and Gaussian elimination, therefore all clauses involving XOR are translated into standard CNF format. Thus, with exactly the same constraints derived in Section 2.5.1, we generate different CNF files encoded by STP and our method, and compare

[4]In 2016, the MILP-based method was applied to the search of differential characteristics and linear trails of Speck [87]. The formulae describing the linear approximations differ from those here, and dedicated techniques are used to improve their search. In addition, the authors concatenate two or three shorter linear trails to attack more rounds, while we focus on finding optimal trails in reduced-round primitives.

Table 2.6: Comparison between the runtime of CNF files generated by Section 2.5.1 and STP on the searching problems of Speck128.

| | Section 2.5.1 | | STP | |
|---|---|---|---|---|
| #Rounds | time1 | time2 | time1 | time2 |
| 4 | 0.05 s | 0.09 s | 2 s | 2 s |
| 5 | 0.8 s | 1 s | 4 s | 7 s |
| 6 | 8 s | 10 s | 18 s | 19 s |
| 7 | 4 m 44 s | 1 m 56 s | 6 m 2 s | 4 m 20 s |
| 8 | 2 s | 643 m 55 s | 55 m 4 s | 114 m 26 s |
| 9 | 53 m 51 s | 16523 m | 10 m 27 s | 12184 m |

their performances on the searching problem of Speck by considering the number of variables and clauses in the corresponding CNF file, as well as the runtime for getting optimal linear trails and *unsatisfiable* decision. Both CNF files run on Cryptominisat4.

In most cases, the CNF file encoded by our method has a smaller number of variables and clauses than the STP-generated ones, and the difference can be a factor of 2 for problems in Speck with larger block sizes. Although the size of the problem and the speed of solving are not strictly proportional, in general, less variables and clauses are preferable. Table 2.6 shows the comparison between the runtime of CNF files generated by the method in Section 2.5.1 and the STP solver, where time1 is the time to find an optimal linear trail, and time2 is the time to return *unsatisfiable*. In general, the performance of both methods is comparable. However it is interesting to notice that it takes 2 seconds to find one optimal trail for 8-round Speck128 by our method while STP uses around one hour. It shows that the performance of CNF files depends heavily on the encoding method and the underlying problem. Therefore, our method may provide an alternative way to solve problems which are not solvable using other solvers.

## 2.5.2 Automatic Search for RX-characteristics

### Translating RX-cryptanalysis with SAT/SMT

Based on the theory of RX-cryptanalysis in Section 2.3, our goal is to find good RX-characteristics in Speck. Different from linear cryptanalysis in Section 2.5.1, we work with related keys with given RX-differences, such that the RX-differences of the plaintext part may somehow be cancelled which leads to distinguishers covering more rounds. For each version of Speck, we model

the propagation of RX-differences through both the round function and the key schedule. Since SPECK uses a non-linear key schedule, the master keys following an RX-characteristic over the key schedule belong to a weak-key class.

We explain the notation as shown in Figure 2.4.

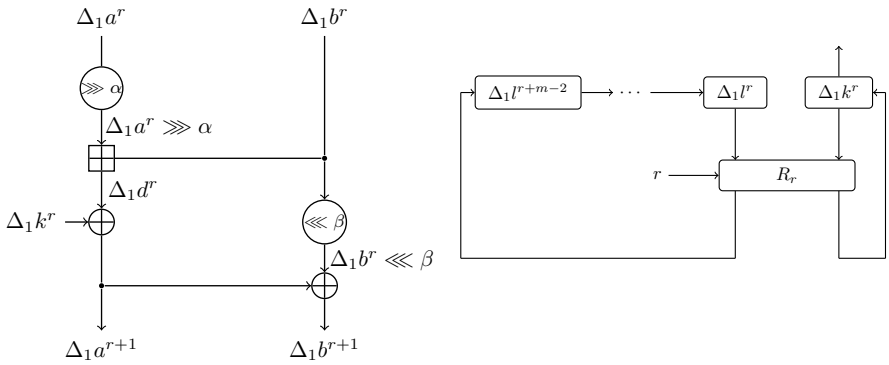



Figure 2.4: Notation of the RX-differences in SPECK. Left: Round function. Right: Key schedule.

Since the key schedule of SPECK reuses the same round function as the cipher itself, it is sufficient to only show the model of the round function. With Corollary 1, the probability of an RX-difference propagating through additions splits into two mutually exclusive cases, thus one of the following constraints should be satisfied:

$$(\mathrm{Id} \oplus \mathrm{SHL})(L'(\Delta_1 a^r \ggg \alpha) \oplus L'(\Delta_1 b^r) \oplus L'(\Delta_1 d^r)) \oplus 1 \preceq \mathrm{SHL}((L'(\Delta_1 a^r \ggg \alpha) \oplus L'(\Delta_1 d^r))|(L'(\Delta_1 b^r) \oplus L'(\Delta_1 d^r))) \tag{2.17}$$

$$(\mathrm{Id} \oplus \mathrm{SHL})(L'(\Delta_1 a^r \ggg \alpha) \oplus L'(\Delta_1 b^r) \oplus L'(\Delta_1 d^r)) \preceq \mathrm{SHL}((L'(\Delta_1 a^r \ggg \alpha) \oplus L'(\Delta_1 d^r))|(L'(\Delta_1 b^r) \oplus L'(\Delta_1 d^r))) \tag{2.18}$$

The cost $w^r$ is calculated as

$$w^r = \begin{cases} |\,\mathrm{SHL}((L'(\Delta_1 a^r \ggg \alpha) \oplus L'(\Delta_1 d^r))|(L'(\Delta_1 b^r) \oplus L'(\Delta_1 d^r)))| + 3\,, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{Eq. (2.17) holds,} \\ |\,\mathrm{SHL}((L'(\Delta_1 a^r \ggg \alpha) \oplus L'(\Delta_1 d^r))|(L'(\Delta_1 b^r) \oplus L'(\Delta_1 d^r)))| + 1.415\,, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{Eq. (2.18) holds.} \end{cases}$$

The linear operations are modelled as follows:

$$\Delta_1 a^{r+1} = \Delta_1 d^r \oplus \Delta_1 k^r,$$

$$\Delta_1 b^{r+1} = (\Delta_1 b^r \lll \beta) \oplus \Delta_1 a^{r+1}.$$

And our objective function is to minimise

$$\sum_r w^r.$$

Starting from Figure 2.4 each operation is replaced with the appropriate constraint(s). This is repeated for each round of the round-reduced cipher, where the output constraints of a round are treated as the input constraints of the next one. A target value is set for the objective function and the program is given as input to the STP tool [89] which searches for a solution satisfying all constraints. When the STP tool finishes, the target value is replaced with a new one according to the search strategy, and the STP tool is called again until the search is complete.

### Search Strategy

The strategy here is based on the binary-search strategy in Section 2.4.3, which works in two phases:

**Phase 1 - finding a good RX-characteristic over the data part.** The program starts by searching for an RX-characteristic covering the data part of the cipher (*i.e.*, the left side of Figure 2.4) with probability at least $2^{-n/2}$, and the key schedule part with probability at least $2^{-mn}$. If a solution satisfying these constraints is found, the objective function for the data part is updated and an RX-characteristic with probability at least $2^{-n/4}$ is searched.

If the program cannot find a solution with probability at least $2^{-n/2}$, the objective function for the data part is relaxed and the program searches for an

**Algorithm 4** Find an optimal RX-characteristic of $r$ rounds for SPECK32/64.

**Input:** $T_d^+, T_d^-, T_k^+, T_k^-$.
**Output:** The probability of an optimal RX-characteristic of $r$ rounds.

$T_d^+ \leftarrow 32$, $T_d^- \leftarrow 0$, $T_k^+ \leftarrow 64$, $T_k^- \leftarrow 0$
Set $T_d^- \leq W_d \leq T_d^+$, $T_k^- \leq W_k \leq T_k^+$
**while** $T_d^+ \neq T_d^-$ **do**
    **if** The problem is *satisfiable* **then**
        $T_d^+ \leftarrow T_d^+/2$
    **else**
        $T_d^- \leftarrow T_d^+/2$
    Set $T_d^- \leq W_d \leq T_d^+$
**while** $T_k^+ \neq T_k^-$ **do**
    **if** The problem is *satisfiable* **then**
        $T_k^+ \leftarrow T_k^+/2$
    **else**
        $T_k^- \leftarrow T_k^+/2$
    Set $T_k^- \leq W_k \leq T_k^+$
**return** $2^{-W_d}, 2^{-W_k}$

RX-characteristic with probability at least $2^{-1.5n/2}$. This binary search (over the binary logarithm of the differential probability of a characteristic in the data part) is repeated until no further improvements are possible.

**Phase 2 - optimising the size of the weak-key class.** After the RX-characteristic with optimal probability is found, the program sets to optimise the size of the weak-key class. Suppose $\zeta_0$ is the probability for the RX-characteristic found in Phase 1, the objective functions in Phase 2 are set such that the program finds RX-characteristics with probability at least $\zeta_0$ for the data part, and probability at least $2^{-mn/2}$ for the key schedule (*i.e.*, the right part of Figure 2.4). With a binary search, the best RX-characteristic for the key schedule is improved under the constraint that this RX-characteristic can support an RX-characteristic for the data part with probability at least $\zeta_0$.

Using this algorithm it is guaranteed that the RX-characteristic in the round function have optimal probability, and that the corresponding RX-characteristic in the key schedule allows for a non-empty weak key class. The algorithm is more formally described in Algorithm 4.

Table 2.7: RX-characteristics with $\gamma = 1$ for different versions of SPECK. Entries marked with † were found through the adjusted search strategy.

| Version | Rounds | Data Prob. | Key Class Size |
|---|---|---|---|
| 32/64 | 10 | $2^{-19.15}$ | $2^{28.10}$ |
| 32/64 | 11 | $2^{-22.15}$ | $2^{18.68}$ |
| 32/64 | 12 | $2^{-25.57}$ | $2^{4.92}$ |
| 48/96 | 11 | $2^{-23.15}$ | $2^{14.93}$ |
| 48/96 | 11† | $2^{-24.15}$ | $2^{25.68}$ |
| 48/96 | 12 | $2^{-26.57}$ | $2^{27.5}$ |
| 48/96 | 12† | $2^{-26.57}$ | $2^{43.51}$ |
| 48/96 | 13 | $2^{-31.98}$ | $2^{24.51}$ |
| 48/96 | 14 | $2^{-37.40}$ | $2^{0.34}$ |
| 48/96 | 15 | $2^{-43.81}$ | $2^{1.09}$ |

### Additional Search Strategies

Note that, for obtaining a large number of rounds, the above search strategy prefers RX-characteristics with high probability in the data part over large weak-key classes. Some may adopt different trade-offs, which can be obtained by minor modifications to the code. For instance, the size of the weak-key class of some characteristics found under our strategy seems to be marginal. We run several experiments with the adjusted search strategy that $2^{mn} \cdot \zeta_0 > 2^{2n}$ where $\zeta_0$ is as before. In such case, the size of the weak-key class is larger than the required data complexity.

### Results

We present an overview of the distinguishers in Table 2.7; more details can be found in our paper [146]. The rotational amount is set to 1 for all the RX-characteristics.

Table 2.8 shows the RX-characteristics covering 11 and 12 rounds found by our program. The best published characteristic so far covered 9 rounds of SPECK with probability $2^{-30}$. Our 10-round characteristic has a much better probability of $2^{-19.15}$ for a weak-key class of size $2^{28.10}$. The table also shows that even our 12-round characteristic has probability of $2^{-26.57}$ which is still higher than the previously known 9-round differential characteristic, although our distinguisher works for a weak-key class of about 30 keys.

Table 2.8: A 11-round (left) and 12-round (right) RX-characteristic in Speck32/64.

| R | RX-diff.(Key) | RX-diff.(data) | R | RX-diff.(Key) | RX-diff.(data) |
|---|---|---|---|---|---|
| 0 | 0000 | (0000\|\|0000) | 0 | 0000 | (0050\|\|2000) |
| 1 | 0000 | (0000\|\|0000) | 1 | 0100 | (8000\|\|0000) |
| 2 | 0000 | (0000\|\|0000) | 2 | 0001 | (0000\|\|0000) |
| 3 | 0001 | (0000\|\|0000) | 3 | 0000 | (0000\|\|0000) |
| 4 | 0000 | (0000\|\|0000) | 4 | 0001 | (0000\|\|0000) |
| 5 | 0003 | (0000\|\|0000) | 5 | 0000 | (0000\|\|0000) |
| 6 | 0200 | (0000\|\|0000) | 6 | 0001 | (0000\|\|0000) |
| 7 | 0205 | (0200\|\|0200) | 7 | 0200 | (0000\|\|0000) |
| 8 | 0801 | (0000\|\|0800) | 8 | 0206 | (0200\|\|0200) |
| 9 | 2001 | (0000\|\|2000) | 9 | 0800 | (0000\|\|0800) |
| 10 | AA0B | (0000\|\|8000) | 10 | 2001 | (0000\|\|2000) |
| 11 | | (2A0B\|\|2A09) | 11 | A40E | (0000\|\|8000) |
| | | | 12 | | (240E\|\|240C) |
| Prob. | $2^{-45.32}$ | $2^{-22.15}$ | | $2^{-59.08}$ | $2^{-25.57}$ |

We extended our search to 13-round characteristics but found none, suggesting that a 12-round RX-characteristic is the longest possible one.

We found RX-characteristics covering up to 15 rounds for Speck48/96. Some of the characteristics are shown in Table 2.9 and Table 2.10. The distinguishers extend the previously best differential characteristic which covers 11 rounds with probability $2^{-45}$. Note that the sizes of the weak key class for the 14- and 15-round characteristics are marginal. However, due to resource constraints we killed the program before it completed its search. Hence, the characteristics presented in this section are not guaranteed to be optimal in length (*i.e.*, 16-round RX-characteristics may exist) nor in probability (*i.e.*, RX-characteristics with higher probabilities or a larger weak-key class may exist for the same number of rounds). In addition, the probabilities of the round function part in the 14- and 15-round characteristics are relatively high, which may imply that distinguishers with larger weak key classes can be found with a different trade-off.

**Experimental Verification.** The characteristics for Speck32/64 were partially verified by experiments. For 10-round and 11-round characteristics, a key-pair is generated randomly with the corresponding difference. Then we executed the key expansion algorithm and tested whether the key characteristic is followed.

Table 2.9: 12-round (left) and 13-round (right) RX-characteristics in SPECK48/96.

| R | RX-diff.(Key) | RX-diff.(data) | R | RX-diff.(Key) | RX-diff.(data) |
|---|---|---|---|---|---|
| 0 | 000008 | (000000\|\|000008) | 0 | 000008 | (000000\|\|000008) |
| 1 | 000240 | (000000\|\|000040) | 1 | 000240 | (000000\|\|000040) |
| 2 | 000000 | (000200\|\|000000) | 2 | 000000 | (000200\|\|000000) |
| 3 | 000000 | (000000\|\|000000) | 3 | 000000 | (000000\|\|000000) |
| 4 | 000000 | (000000\|\|000000) | 4 | 000000 | (000000\|\|000000) |
| 5 | 000000 | (000000\|\|000000) | 5 | 000000 | (000000\|\|000000) |
| 6 | 000001 | (000000\|\|000000) | 6 | 000001 | (000000\|\|000000) |
| 7 | 000001 | (000000\|\|000000) | 7 | 000001 | (000000\|\|000000) |
| 8 | 000001 | (000000\|\|000000) | 8 | 000000 | (000000\|\|000000) |
| 9 | 010010 | (000001\|\|000001) | 9 | 010018 | (000001\|\|000001) |
| 10 | 100089 | (000010\|\|000018) | 10 | 1000F1 | (000018\|\|000010) |
| 11 | 8904DE | (000080\|\|000040) | 11 | 880801 | (080080\|\|080000) |
| 12 | | (09049E\|\|09069E) | 12 | C04911 | (000000\|\|400000) |
| | | | 13 | | (004911\|\|004913) |
| Prob. | $2^{-52.49}$ | $2^{-26.57}$ | | $2^{-71.49}$ | $2^{-31.98}$ |

Once a weak key was found, we encrypted $2^{32}$ plaintexts, and measured the probability that the RX-characteristic is satisfied. It shows that the experimental probability matches the theoretical prediction.

## 2.6 Conclusion

In this chapter, we recalled the cryptanalytic techniques of ARX ciphers, including differential cryptanalysis, linear cryptanalysis, rotational cryptanalysis, as well as our recently-proposed rotational-XOR cryptanalysis. We studied the properties of the rotational-XOR difference with a rigorous mathematical deduction, which enrich the toolbox for analysing ARX ciphers. For various cryptanalytic techniques, a core element is the distinguisher. Generally speaking, it is difficult to find a good distinguisher for a cipher with a large block size and many rounds, especially if one also prefers an optimal solution. As an application of a general framework with SAT/SMT solvers, we developed an automatic tool for finding optimal characteristics and trails in ARX ciphers. Moreover, a Python-based package SYMPY was adopted to wrap the search tool into ARXPY. It bridges plain implementations of ARX ciphers and SAT/SMT files, thus simplifies the cryptanalysis. With the automatic search tool, the effort

Table 2.10: 14-round (left) and 15-round (right) RX-characteristics in Speck48/96.

| R | RX-diff.(Key) | RX-diff.(data) | R | RX-diff.(Key) | RX-diff.(data) |
|---|---|---|---|---|---|
| 0 | 000008 | (000000\|\|000008) | 0 | 000008 | (000000\|\|000008) |
| 1 | 000240 | (000000\|\|000040) | 1 | 000240 | (000000\|\|000040) |
| 2 | 000000 | (000200\|\|000000) | 2 | 000000 | (000200\|\|000000) |
| 3 | 000000 | (000000\|\|000000) | 3 | 000000 | (000000\|\|000000) |
| 4 | 000000 | (000000\|\|000000) | 4 | 000000 | (000000\|\|000000) |
| 5 | 000000 | (000000\|\|000000) | 5 | 000000 | (000000\|\|000000) |
| 6 | 000001 | (000000\|\|000000) | 6 | 000001 | (000000\|\|000000) |
| 7 | 000001 | (000000\|\|000000) | 7 | 000001 | (000000\|\|000000) |
| 8 | 000000 | (000000\|\|000000) | 8 | 000001 | (000000\|\|000000) |
| 9 | 010018 | (000000\|\|000000) | 9 | 010011 | (000001\|\|000001) |
| 10 | 1000E0 | (010019\|\|010019) | 10 | 100080 | (000010\|\|000018) |
| 11 | 680021 | (0801E8\|\|000120) | 11 | 990391 | (000089\|\|000049) |
| 12 | 000009 | (000900\|\|000000) | 12 | 480103 | (000248\|\|000000) |
| 13 | 202844 | (000000\|\|000000) | 13 | 000301 | (000100\|\|000100) |
| 14 | | (202844\|\|202844) | 14 | 91101D | (000000\|\|000800) |
| | | | 15 | | (91181D\|\|91581D) |
| Prob. | $2^{-95.66}$ | $2^{-37.40}$ | | $2^{-94.91}$ | $2^{-43.81}$ |

in finding distinguishers in ARX with differential, linear or RX-cryptanalysis will be largely reduced, therefore, it allows us to have a better understanding of an ARX cipher or to design new ones.

Our automatic search tool is mainly applied to linear cryptanalysis and RX-cryptanalysis of Speck. Here, we list the best known distinguishers of Speck32 and Speck48 in Table 2.11 including differential characteristics, linear trails and RX-characteristics.

Considering the number of rounds covered by the distinguishers, RX-characteristics are better than differential characteristics and linear trails within some weak key class. For the smallest version Speck32/64, the distinguishers cover up to 54.5% of the total rounds, and the best known attack reached 15 rounds with differential cryptanalysis by Dinur [74]. Interestingly, the distinguishers in the larger versions cover many more rounds while the total number of rounds only slightly increases. Meanwhile, the distinguishers we found for the larger versions are often not optimal due to the resource limits. It implies that with improved computational resources one might find better distinguishers with our automatic search tool for the larger versions of Speck.

Table 2.11: Best found distinguishers for Speck32 and Speck48. DC: differential characteristic; LC: linear trail; RX: RX-characteristic. The data probability for a linear trail is filled with its squared correlation.

| Version | Rnds. | Covered. Rnds. | Type | Data Prob. | Key Class | Ref. |
|---|---|---|---|---|---|---|
| 32/64 | 22 | 9 | DC | $2^{-30}$ | $2^{64}$ | [74] |
| | | 9 | LC | $2^{-28}$ | $2^{64}$ | [150] |
| | | 10 | RX | $2^{-19.15}$ | $2^{28.10}$ | [146] |
| | | 11 | RX | $2^{-22.15}$ | $2^{18.68}$ | [146] |
| | | 12 | RX | $2^{-25.57}$ | $2^{4.92}$ | [146] |
| 48/96 | 23 | 10 | DC | $2^{-40}$ | $2^{96}$ | [74] |
| | | 10 | LC | $2^{-44}$ | $2^{96}$ | [150] |
| | | 11 | DC | $2^{-45}$ | $2^{96}$ | [87] |
| | | 11 | RX | $2^{-24.15}$ | $2^{25.68}$ | [146] |
| | | 12 | RX | $2^{-26.57}$ | $2^{43.51}$ | [146] |
| | | 13 | RX | $2^{-31.98}$ | $2^{24.51}$ | [146] |
| | | 14 | RX | $2^{-37.40}$ | $2^{0.34}$ | [146] |
| | | 15 | RX | $2^{-43.81}$ | $2^{1.09}$ | [146] |

More importantly, the larger versions in the family are those which received more attention and consideration, so it is certainly worth a closer look before any decision being made for recommendation or application.

In addition, we expect the applications of our analysis tool to many other ARX designs in future work. Moreover, it is certainly promising to design new ARX primitives efficiently by automated evaluations and adjustments. To this end, further improvements in automatic search techniques are required.

# Chapter 3

# Optimised Interpolation Attacks on LowMC

## 3.1 Motivation

With the concepts of secure multi-party computation (MPC) and fully homomorphic encryption (FHE) receiving increasing attention, these areas witnessed remarkable advances in the past years, due to a number of theoretical and practical breakthroughs. In MPC protocols, the communicating parties evaluate a function collaboratively with the inputs being private, where Yao's circuit [210] and Shamir's secret sharing [176] are the main practical protocols. Symmetric-key primitives are not directly involved in the techniques that enable the computation by multiple parties. However, a secure evaluation of a symmetric encryption, such as an encryption with the AES, is a vital benchmark for an MPC protocol. For fully homomorphic encryption, a user may want to upload the encrypted information to a cloud server, while the server is still able to carry out certain operations on the ciphertexts such that the decryption yields the intended evaluation on the plaintext. Instead of sending a homomorphic encryption of the information under the public key to the server, the user could send the symmetric-key encryption of the message together with the public-key encryption of the secret key. Then, the server could use a homomorphic decrypting circuit to get the homomorphic encryption of the message while avoiding heavy computations. One of the practical difficulties is that, highly-optimised implementations of the existing cryptographic primitives are not directly transplantable to MPC and FHE with efficient instantiations.

For both MPC and FHE scenarios, there are two main metrics to evaluate the cost of block ciphers: multiplicative complexity (MC) which is simply the number of multiplications (AND gates) in a circuit and the multiplicative depth of the circuit (AND depth). The metrics have some similarity to the latency, where a low-latency cipher takes less time for one encryption in hardware. It also resembles the efficient masked implementations which require minimising the number of multiplications. The requirements from the MPC and FHE applications hint a new direction in designing lightweight block ciphers instead of optimising ciphers with a different nature. Proposed by Albrecht *et al.* in 2015, LowMC is the first block cipher that is designed for low multiplicative complexity. Since 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 generated by random invertible matrices with partial non-linear layers where the S-boxes are of algebraic degree 2. The design rationale and security are carefully analysed to support the security claim of the proposal, where a formula was derived to estimate the required number of rounds in order to resist classical cryptanalytic techniques.

In this chapter, we consider optimised interpolation attacks on the first block cipher for low multiplicative complexity, LowMC. First, we recall the main methodology and the cipher specification in Section 3.2 and Section 3.3. Our basic attack on 9-round LowMC with an 80-bit key is described in Section 3.4. A generic framework for optimised interpolation attacks is described in Section 3.5 with an application to 9-round LowMC-80. Finally, we conclude in Section 3.6.

## 3.2 Higher-Order Differential Cryptanalysis and Interpolation Attacks

Higher-order differential and interpolation attacks are examples of algebraic cryptanalysis which explore certain properties in the algebraic normal form (ANF) of the outputs as Boolean polynomials. Note that there is a specific attack in cryptanalysis called algebraic attack [60], which builds and solves equation systems with known information from the inputs and outputs, the unknown variables, *i.e.*, the secret keys. Solving such equation systems is difficult in general [52]. In this chapter, we refer to algebraic cryptanalysis in a general sense, which differs from the algebraic attack.

First, we recall the basic concepts and introduce the notation used in this chapter. Any function $F$ from $\mathbb{F}_{2^n}$ to $\mathbb{F}_2$ can be described as a multivariate

polynomial, whose ANF is unique and given as

$$F(x_{n-1},\ldots,x_0) = \sum_{u=(u_{n-1},\ldots,u_0)} \alpha_u M_u\,,$$

where $\alpha_u \in \{0,1\}$ is the coefficient of the monomial $M_u = \prod_{i=0}^{n-1} x_i^{u_i}$, and the sum is over $\mathbb{F}_2$. The algebraic degree of the function $F$ is $\deg(F) = \max\{wt(u) \,|\, \alpha_u \neq 0\}$. 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 coefficients $\alpha_u$ of $F$ can be interpolated by summing 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 are a subset of the bits set to 1 in $u_{n-1},\ldots,u_0$. More formally, we have the following lemma.

**Lemma 4.** *Let $S = \{x = (x_{n-1},\ldots,x_0) \,|\, \bar{u} \wedge x = 0\}$, where $\bar{u}$ is bitwise NOT applied to $u$ and $\wedge$ is bitwise AND. Then, $\alpha_u = \sum_{(x_{n-1},\ldots,x_0)\in S} F(x_{n-1},\ldots,x_0)$.*

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 $\{x = (x_{n-1},\ldots,x_0) \,|\, wt(x) \leq d\}$.

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 $\alpha_u$. This ANF representation can be efficiently computed using the *Möbius Transform*, which is an FFT-like algorithm. The Möbius 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 Möbius transform, we refer to Joux [112].

**Higher-Order Differential Cryptanalysis.** Higher-order differential cryptanalysis was introduced by Lai in [131] as an algebraic-type cryptanalysis that is particularly efficient against ciphers of a low algebraic degree. The name stems from an extension of the differences (as first-order derivatives) called *higher-order derivatives*.

**Definition 11.** *Let $(S,+)$ and $(T,+)$ be Abelian groups. For a function $f : S \to T$, the $i$-th derivative of $f$ at point $(a_1, a_2, \cdots, a_i)$ is recursively defined as*

$$D^{(i)}_{a_1,\cdots,a_i} f(x) = D_{a_i}(D^{(i-1)}_{a_1,\cdots,a_{i-1}} f(x))\,,$$

*where the first derivative at point a, $D_a^{(1)}$ (or $D_a$ for simplicity) is defined by $f(x+a) - f(x)$, with $t_1 - t_2 = t_1 + t_2^{-1}$, $t_1, t_2 \in T$.*

It is obvious that the 0-th derivative is $f(x)$ itself, and the first-order derivative is simply the difference in differential cryptanalysis according to Definition 4 in Chapter 2. A higher-order differential distinguisher distinguishes a function $f$ from random if

$$D^{(i)}_{a_1,\cdots,a_i} f(x) = constant\,,$$

for some $i$-th order derivative. The higher-order differential property is closely related to the algebraic degree of a function. The basic higher-order differential cryptanalysis over $\mathbb{F}_2$ considers some target bit $b$ and analyses 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, the attacker chooses an arbitrary linear subspace $S$ of dimension $dg$, and evaluates the cipher over its $2^{dg}$ 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 higher-order differential over $S$ for the target bit $b$ is equal to some constant. Higher-order differential properties may be used in key recovery attacks, depending on the specification of the cipher [124].

It is worthy of mentioning that higher-order differential cryptanalysis resembles several other cryptanalytic methods, including integral cryptanalysis [122], zero-sum distinguisher [11], and cube attack [76], because they all focus on the algebraic degree of the ANF or some monomials. For instance, an attacker asks for the sum of the encryptions over a plaintext subspace: if the sum is zero for all keys, then an integral distinguisher is found. In such a case, the output bits are called *balanced*.

**Interpolation Attacks.** The interpolation attack was introduced in 1997 by Jakobsen and Knudsen as an algebraic-type attack on block ciphers [111]. The attack is closely related to higher-order differential cryptanalysis and is particularly efficient against block ciphers whose round function has a low algebraic degree.[1] 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 $\mathbb{F}_2$, 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 ciphertext side in terms of the ciphertext and key as $F(C, K)$. The key $K$ is an unknown constant,

[1] In fact, some of its variants directly exploit higher-order differential properties, as we describe next.

and we can write $F_K(C) = F_K(c_{n-1}, \ldots, c_0) = \sum_{u=(u_{n-1},\ldots,u_0)\in\mathbb{F}_{2^n}} \alpha_u M_u$, where $\alpha_u \in \{0,1\}$ is the coefficient of the monomial $M_u = \prod_{i=0}^{n-1} c_i^{u_i}$. Therefore, the coefficients $\alpha_u$ of $F_K(C)$ generally depend on the secret key and are unknown. The goal of the interpolation attack is to recover (interpolate) the unknown coefficients of $F_K(C)$, and then use various ad hoc techniques to recover the secret key.

In order to deduce the unknown coefficients of $F_K(C)$, they are considered as linearised variables, 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 plaintext bits is relatively small, which allows to use higher-order differential cryptanalysis. More specifically, a higher-order differential property is devised by encrypting a subspace $S$ of plaintexts of dimension $dg+1$, and performing higher-order differentiation with respect to this subspace, whose outcome is zero on the bit $b$.

When expressed in terms of the ciphertexts $C_1, \ldots, C_{2^{dg+1}}$, this gives the equation $\sum_{t=1}^{2^{dg+1}} F_K(C_t) = 0$. For each ciphertext $C_t$, $F_K(C_t)$ is merely a linear expression in the variables $\alpha_u$, and thus the subspace $S$ gives rise to one linear equation in the variables $\alpha_u$. In order to solve for the unknown variables $\alpha_u$, the attacker considers several such subspaces, each giving one equation. In total, the number of equations needs to be roughly equal to the number of the unknown $\alpha_u$ variables, assuming the equations are sufficiently "random". The efficiency of the attack depends on the algebraic degree of $b$ in terms of the plaintext, but also on the number of unknown coefficients in the ANF representation of $b$ in terms of the ciphertext.

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 $\alpha_u$. Therefore, in order to mount efficient interpolation attacks, the attacker tries to minimise these parameters, since a straightforward application will encounter complexity obstacles within only a few rounds. To this end, we had to develop new techniques such as using carefully chosen plaintext structures which allow to efficiently derive the linear system of equations. Our main new contribution is based on considering two variants of the interpolation attack, as illustrated in the following example.

In the original variant of the interpolation attack over $\mathbb{F}_2$ which we refer to as variant 1, the attacker views the ANF of some intermediate encryption bit $b$ as an unknown polynomial $F_K(C)$ in the ciphertext bits, where $K$ is the unknown but fixed secret key. In a dual approach to the interpolation attack, which we refer to as variant 2 proposed by Shimoyama *et al.* [178], 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 variable. For example, consider the polynomial

$$F(c_1, c_2, x_1, x_2, x_3) = c_1c_2x_1 + c_1c_2x_2 + c_1x_1 + c_1x_2 + c_2x_1 + x_1x_2 + x_3 + 1,$$

where $x_1, x_2, x_3$ are the key bits and $c_1, c_2$ are the ciphertext bits. 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 as 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 because it requires evaluating simple monomials in the ciphertext. Therefore, the choice of which variant to use in order to optimise the attack depends on the underlying cryptosystem.

As detailed in Section 3.5, 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 [111]. Then, we transform a carefully chosen variable subset to variables which are linearised 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 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: $x_1, x_2, \alpha_4$. 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.

## 3.3 Description of LowMC

LowMC is a family of SPN ciphers, proposed at Eurocrypt 2015 [3] by Albrecht *et al.* The specification defined two specific instance families which are analysed

Table 3.1: LowMC Instance Families

| Instance | key size $\kappa$ | block size $n$ | S-boxes $m$ | data $lim$ | rounds $r$ |
|---|---|---|---|---|---|
| LowMC-80 | 80 | 256 | 49 | 64 | 11 |
| LowMC-128 | 128 | 256 | 63 | 128 | 12 |

in this chapter, both having a block size of $n = 256$ bits, and are characterised by their key size $\kappa$, which is either 80 or 128 bits. 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 round-key addition layer, an S-box layer, and an affine layer over $\mathbb{F}_2$. 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. To be specific, the linear operations are defined by random invertible binary matrices. 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 $n$. The S-box layer of LowMC is composed of 3-bit S-boxes with degree 2 over $\mathbb{F}_2$ with each S-box being $S[8] = \{0, 1, 3, 6, 7, 4, 5, 2\}$. Furthermore, the S-box layers are only partial, namely, in each S-box layer, only $3m < n$ bits go through an S-box (where $m$ is a parameter), while the rest of the $n - 3m$ bits remain unchanged.

The distinctive feature of LowMC is that its affine layers are chosen at random and thus each block cipher family contains a huge number of instances. This may enable a malicious party to instantiate LowMC with a hidden backdoor. Its designers propose to use the Grain stream cipher [105] as a source of pseudo-random bits in order to restrict that 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 designers proposed that each family instance of LowMC is defined with a data limit $lim$, which determines the maximal 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 3.1. 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 cryptanalysis. Indeed, LowMC is potentially susceptible to such 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. All of our results were obtained using the interpolation attack.

To simplify the notations, 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 S-box 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 S-boxes in the S-box 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-part of the state, respectively (*e.g.*, $Y^5|IP$ is the I-part of the input state to the 6-th S-box layer).

It is common in cryptanalysis of block ciphers to exchange the order of the final two affine operations over $\mathbb{F}_2$, 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 did already "peel 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.

**Model of Computation** Since an exhaustive key search attack and our attacks use different bitwise operations, comparing these attacks cannot be compared 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. Each affine layer of LowMC involves multiplication of the 256 state with a $256 \times 256$ matrix. This multiplication requires roughly $2^{16}$ bit operations, and therefore a single encryption of LowMC requires more than $2^{16} \cdot 8 = 2^{19}$ bit operations. Consequently, an 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.

## 3.4 A Basic 9-Round Attack on LowMC-80

In this section we describe our basic interpolation attack on 9-round LowMC. We begin by considering the elements that are required for the attack.

### 3.4.1 The Higher-Order Differential Property

We construct the higher-order differential property used in the interpolation attack. A similar property was described by the LowMC designers [3], but we reiterate it here for the sake of completeness.

The algebraic degree of a single round of LowMC-80 over $\mathbb{F}_2$ is 2, and therefore the algebraic degree of any bit at the input to the 6-th S-box 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 S-boxes in the first round, the degree at the input to the 7-th S-box layer, $Y^6$, in the bits of the I-part, $X^0|IP$, is at most 32. Furthermore, since the bits of the I-part of the 7-th S-box layer do not go through an S-box, the degree of any bit of $Z^6|IP$ in the input bits of the I-part, $X^0|IP$, is at most 32.

The last properties implies that the value of a 33rd order differential over any 33-dimensional subspace selected from $X^0|IP$ is zero for any bit of $Z^6|IP$. Moreover, as we selected a subspace whose bits do not go through an S-box in the first round, the value of a 32nd order differential for any bit of $Z^6|IP$ over any 32-dimensional subspace from $X^0|IP$, is a constant and 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 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.

### 3.4.2 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 have $F_K(C) = \sum_{u \in \mathbb{F}_{2^{256}}} \alpha_u M_u$, where $\alpha_u \in \{0, 1\}$ is the coefficient of the monomial $M_u = \prod_{i=0}^{255} c_i^{u_i}$. As the complexity of the attack depends on the number of variables $\alpha_u$, 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 two, and thus $\deg(F_K(C)) \leq 4$. This implies that $\alpha_u = 0$ 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 $\alpha_u$ according to the degree of $M_u$, by defining the set of variables $U_i$ for a positive integer $i$ as $U_i = \{\alpha_u \text{ that is not identically zero as a function of the key} | wt(u) = i\}$. We have already seen that $U_i$ is empty for $i > 4$, and we now derive tighter bounds on $U_i$ for $i \leq 4$. Thus, we analyse 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.

The ciphertext $Y^9$ contains 256 bits, 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 S-box layer is applied to $Z^8$ to obtain the state $Y^8$. Each 3-bit S-box may contribute up to 3 quadratic monomials to $Y^8$, and 6 monomials in total, *e.g.*, an S-box 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 $\alpha_u$ of every monomials $M_u$ in the ciphertext is linearised 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 S-boxes, the total number of monomials $M_u$ in the ANF of the state bits of $Y^8$ is bounded by $|U_2| \leq 3 \cdot 49 = 147$, $|U_1| \leq 256$ 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 S-box 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 S-box layer is of degree 2, a trivial upper bound on the number of variables $\alpha_u$ 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

$\{u|\alpha_u \in U\}$ can be easily derived during preprocessing, which involves a more explicit computation of the monomial set $\{M_u|\alpha_u \in U\}$.

### 3.4.3 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 approach 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 [76] 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 by 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 $S$.

### 3.4.4 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 $\alpha_u$ using $e \approx |U|$ equations, without recovering the actual secret key. The details of this final step will be given in the optimised attack in Section 3.5. For the sake of convenience, we describe the attack in two phases: the preprocessing phase and the 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 $\hat{a}_0$: evaluate $b$ on the subset of inputs of $S$ with the key set to zero, and obtain a bit array of size $2^{37}$, then, 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 $\hat{a}_0$.
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 S-box layer, and multiplying them in pairs.

**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 $F_K(C_t)$.
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 $M_u(C_t)$ and set the corresponding bit entry in $A[t]$ according to the result.
4. Allocate an $e \times |U|$ matrix $E$ over $\mathbb{F}_2$, representing the equation system on $U$.
5. For each 32-dimensional subspace $S_j$ in $S$, namely $S_1, \ldots, S_e$:
   (a) Populate the row $E[j]$ by summing over the $2^{32}$ rows of $A$ corresponding to $S_j$.
6. Solve the equation system $E\hat{x} = \hat{a}_0$, where $\hat{x}$ represents the vector of variables of $U$ and $\hat{a}_0$ 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, where 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.[2] 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 in Step 3.a and Step 5. The reason that we can ignore these mappings in the complexity evaluation is that they are independent of the secret key and data, so they can be precomputed.

[2]The complexity of solving linear equation system is not the dominating phase in the total time complexity in our attacks. Even though advanced algorithms such as Strassen's algorithm [185] can be found in the literature, requiring a complexity of about $O(n^{2.8})$, they are very complex and inefficient in practice comparing with the Gaussian elimination algorithm which is $O(n^3)$ in complexity.

## 3.5 The Optimised Interpolation Attack

In this section, we introduce three optimisations of the basic 9-round attack above. The first optimisation reorders the steps of the algorithm in order to reduce the memory complexity, while the second optimisation further exploits the structure of chosen plaintexts to reduce the time complexity of the attack. Finally the third optimisation is based on a novel technique in interpolation attacks, and allows to further reduce the data and time complexities.

The first two optimisations 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, $\hat{a}_\ell$) and then populate the corresponding column $E[*][\ell]$ by summing over the 32-dimensional subspaces $S_1, \ldots, S_e$ on $\hat{a}_\ell$.

Next, we reduce the time complexity by optimising the summation process: given a bit array $\hat{a}_\ell$ 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 Möbius transform. For this purpose, we can view $\hat{a}_\ell$ as evaluating a 37-variable polynomial over $\mathbb{F}_2$, and the summation over a 32-dimensional subspace of $\hat{a}_\ell$ is equal to the coefficient of its corresponding 32-degree monomial. All these coefficients are computed by the Möbius transform in about $37 \cdot 2^{37}$ bit operations. We stress that the reason that we can use the Möbius transform in this case is purely combinatorial and is due to the way that we selected the structure of subspaces for the interpolation to $\hat{a}_\ell$ when viewed as a polynomial.

Finally, we optimise the data complexity and further reduce the time complexity. In order to achieve this, examine the polynomial $F(K, C)$ for the target bit $b$ selected in $Z^6|IP$. Due to the linear key schedule of LowMC, this polynomial is of degree 4. We consider a variable $\alpha_u \in U$ and analyse its ANF in terms of the 80 key bit variables. Since $\alpha_u$ is multiplied by $M_u$ 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 [76] and can be used to significantly reduce the number of variables $U$.

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.

Next, we describe the general steps to transform the variable set and the equations, and the full analysis on the details of the algorithms and the deduction

of the complexity are referred to our paper [75].

### 3.5.1 Transformation of Variables

Given an instance of LowMC with a 256-bit block, a key size of $\kappa$, and $m$ S-boxes 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 S-box 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 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,

$$|U| \leq \binom{256 + 3m}{\leq 2^{r_1-1}}.$$

Note again that this bound is generally better than the trivial bound of $|U| \leq \binom{256}{2^{r_1}}$, which follows from 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, $F(K, C) = F(x_{\kappa-1}, \ldots, x_0, c_{255}, \ldots, c_0) = \sum_{u \in \mathbb{F}_{2^n}} \alpha_u M_u$, where $M_u = \prod_{i=0}^{n-1} c_i^{u_i}$ and $\alpha_u(x_{\kappa-1}, \ldots, x_0)$ is a polynomial from $\mathbb{F}_{2^\kappa}$ to $\mathbb{F}_2$. 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 as $\deg(\alpha_u) \leq d - \deg(M_u)$.

In order to estimate the number of variables with improved accuracy, it is preferable to have an adjustable parameter to determine which variables are to be interpolated with monomials involving the key bits. To this end, we choose an integral *splitting index* $1 \leq sp \leq d + 1$, and split the variable set into $U = U' \bigcup U''$, where $U' = \bigcup_{i=1}^{sp-1} U_i$ and $U'' = \bigcup_{i=sp}^{d} U_i$. Therefore, we can interpolate each variable of $U''$ in terms of the key, and express it as $\alpha_u =$

$\sum_{\{v=(v_{\kappa-1},\ldots,v_0)|wt(v)\leq d-sp\}} \beta_v \tilde{M}_v$, where $\beta_v \in \{0,1\}$ is the coefficient of the monomial $\tilde{M}_v = \prod_{i=0}^{\kappa-1} x_i^{v_i}$. Note that the coefficients $\beta_v$ are independent of the key and can be computed during preprocessing. This interpolation transforms the set of variables $U''$ into the set of variables $V = \{\tilde{M}_v = \prod_{i=0}^{\kappa-1} x_i^{v_i} | wt(v) \leq d - sp\}$, which are low degree monomials in the key bits. 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 $V_i = \{\tilde{M}_v \in V | \deg(\tilde{M}_v) = i\}$. Especially, the variables in $V_1$ are exactly the secret key bits. In addition, we define $V_{\leq i} = \bigcup_{j=1}^{i} V_i$. The set of variables is transformed via interpolation into a new set of variables $W = U' \bigcup V$.

The variable $\alpha_u \in U_i$ is interpolated by the variables in $V_{\leq(d-i)}$. This process can be executed in a similar way as the two-phase process in the basic interpolation attack. In the first phase, we evaluate the polynomial $\alpha_u$ for all relevant keys of Hamming weights at most $d - i$. Since $\alpha_u$ is a variable in $U_i$, the evaluation of $\alpha_u$ is simply the summation of the target bit on a subset of $2^i$ inputs. Then, the coefficients $\beta_v$ of the monomials $\tilde{M}_v \in V_{\leq d-i}$ are interpolated by summing the evaluations of $\alpha_u$. In addition, the equation system need to be transformed correspondingly with the new variable set $W$, namely, the variables $\alpha_u \in U_i$ in a linear expansion of the monomials $\tilde{M}_v$. With a linear transformation on the coefficient matrix of the original equation system, it can be converted into a new equation system based on the new set $W$. Therefore, the complexity can be estimated based on the number of coefficients in the linear expansion, which is $|V_{\leq(d-i)}|$.

Here, we give an summary of the optimised interpolation attack.

**Preprocessing:**

1. Compute an $e$-bit array of free coefficients for $e \approx |U|$ equations, denoted by $\hat{a}_0$: 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 Möbius transform to the bit array, and copy the values of sums over $S_1, \ldots, S_e$ to $\hat{a}_0$.
2. Calculate the $|U|$ vectors $\{u|\alpha_u \in U\}$: This is done by first calculating the $256 + 3m$ monomials past the first S-box layer, and multiplying them in unordered tuples of size up to $2^{r_1-1}$ .

**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 $\hat{a}_\ell$ (the evaluation of the $\ell$-th monomial in $U$).
3. Allocate an $e \times |W|$ matrix $E$ over $GF(2)$, representing the 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 $\{M_u | \alpha_u \in U\}$:
   (a) For each ciphertext $C_t$, calculate $\hat{a}_\ell[t]$ by evaluating $M_u(C_t)$.
   (b) Use the Möbius transform to sum over all subspaces of $\hat{a}_\ell$.
   (c) When $\alpha_u \in U'$: For each subspace $S_j$ in $S$, obtain its corresponding sum from $\hat{a}_\ell$ and copy it to the corresponding column of $E_1[j]$.
   (d) Otherwise, when $\alpha_u \in U''$: Interpolate the coefficients $\beta_v$ of $V_{\leq(d-i)}$ in $\alpha_u$. For each subspace $S_j$ in $S$, obtain its corresponding Boolean sum from $\hat{a}_l$ as the coefficient of $\alpha_u$ over $U$. Populate the corresponding column of $E_2[j]$ by adding the sum when $\beta_v$ is 1.
5. Solve the equation system $E\hat{x} = \hat{a}_0$, where $\hat{x}$ represents the vector of variables of $W = U' \bigcup V$.
6. Deduce the $\kappa$-bit secret key, which is simply given by the monomials $V_1$.

The total data complexity of the algorithm is $|S|$ chosen plaintexts. The total time complexity is dominated by Step 4 and 5, where Step 4.b is potentially the main factor with a complexity of $\log(|S|) \cdot |S| \cdot |U|$. 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.

### 3.5.2 Applications to LowMC-80

First of all, we apply the optimised interpolation attack to 9-round LowMC, as a comparison with the basic version attack. As in the attack described in

Section 3.4.4, 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 $|U| = \binom{256+3m}{\leq 2^{r_1-1}} = \binom{403}{\leq 2} \approx 2^{16.5}$.

We use $sp = 2$. The size of the relevant variable sets can be deduced: $|U'| \leq \binom{256}{\leq sp-1} = \binom{256}{\leq 1} \approx 2^8$, $|V| \leq \binom{\kappa}{\leq d-sp} = \binom{80}{\leq 2} < 2^{12}$, $|W| = |U'| + |V| < 2^{12}$.

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. The time complexity of the optimised 9-round attack is about $2^{57}$ bit operations (or $2^{57-19} = 2^{38}$ encryptions), mostly dominated by Step 4.b with $log(|S|)\cdot|S|\cdot|U| \approx 35 \cdot 2^{35} \cdot 2^{16.5} = 2^{56.5}$ operations. 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.

Similar to the 9-round attack, the attack can be directly applied to 10-round LowMC-80 with a finely adjusted splitting index $sp = 4$. For the full LowMC-80, Aa direct application of the optimised attack would lead to an attack slower than exhaustive search, however, the 9-round and 10-round attack can be extended into an attack on the full LowMC by assuming a linear dependency between the bits of $Z^6|IP$ and $Z^7|IP$, each with 109 bits. Since the linear layer is generated randomly, it is plausible to assume the existence of weak instances, where the probability of this event is about $2^{109+109-256} = 2^{-38}$.

## 3.6 Conclusions

In this chapter, we study the resistance of the LowMC cipher against higher-order differential cryptanalysis and the interpolation attack. The low algebraic degree of the partial S-box layer allows us to find higher-order differential distinguishers covering more than half of the full cipher, in addition LowMC it has an inadequate number of monomials in the ANF of an intermediate bit in terms of the ciphertexts and the key, which renders it vulnerable to interpolation attacks. We introduced new techniques for interpolation attacks, including a new variable transformation technique that can lead to savings in their data and time complexities of the attack. We apply the optimised interpolation attack to LowMC-80 reduced to 9 rounds, to demonstrate the improvement over the basic approach.

Furthermore, the optimised interpolation technique can be further applied to full versions of LowMC-80 and LowMC-128 for some weak instances. We skipped the detailed elaborations in this thesis. A summary of the attack results can be found in Table 3.2, which refute the designers' security claim.

Table 3.2: Attacks on LowMC

| Instance Family | Nr. of Rounds | Rounds Attacked | Fraction of Instances | Data† | Time†† | Memory††† |
|---|---|---|---|---|---|---|
| LowMC-80 | 11 | 9 | 1 | $2^{35}$ | $2^{38}$ | $2^{35}$ |
| | | 10 | 1 | $2^{39}$ | $2^{57}$ | $2^{39}$ |
| | | all(11) | $2^{-38}$ | $2^{39}$ | $2^{57}$ | $2^{39}$ |
| LowMC-128 | 12 | 11 | 1 | $2^{70}$ | $2^{86}$ | $2^{70}$ |
| | | all(12) | $2^{-122}$ | $2^{70}$ | $2^{86}$ | $2^{70}$ |
| | | all(12) | 1 | $2^{73}$ | $2^{118}$ | $2^{80}$ |

† Given in chosen plaintexts.
†† Given in LowMC encryptions.
††† Given in 256-bit words.

For instance, it can be shown 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.

In response to our attack, the designers of LowMC have improved the security margin by extending the number of rounds in each version of LowMC, and released an updated cipher LowMCv2 [4]. Following this line of MPC-friendly and FHE-friendly ciphers, several new block ciphers and stream ciphers are proposed for achieving low multiplication complexity and low AND depth, such as MiMC [2, 95], Kreyvium [55], and Flip [155].

As a future work item, it will be interesting to optimise our techniques further and apply them to additional block ciphers.

# Chapter 4

# Observations on Invariant Subspace Attack

Linear layers are important components in designing symmetric-key primitives, for both implementation and security. In terms of security, a widely accepted criterion of a linear layer is its branch number, which is essential to achieve provable security against differential and linear cryptanalysis. The properties of the linear layers are further studied to evaluate the resistance to other cryptanalytic methods. For instance, the resistance of a so-called "structure" to impossible differential cryptanalysis is closely related to the "primitive index" of the linear layer [186]. In this chapter, we study the influence of the linear layers on distinguishers in the invariant subspace attack.

This chapter is organised as follows. In Section 4.1, we recall the invariant subspace attack and mathematically characterise some of the properties. Section 4.2 gives the bounds on invariant subspaces for several block ciphers, in terms of the property of their linear layers, which further results in countermeasures to achieve provable resistance. Finally, we conclude in Section 4.3.

## 4.1 Invariant Subspace Attack

Over the last fifteen years, lightweight block ciphers which are suitable for various constrained environments have become the focus of recent symmetric primitives. Performance always comes with a price. Some lightweight block

ciphers trade in a tolerable amount of security margin under certain attack models to achieve improved performance in hardware. Without an explicit guideline, it is an interesting question whether a slight change towards higher efficiency may have devastating consequences. When choosing lightweight S-boxes, the linear layer and the key schedule (sometimes even no key schedule), the security margin of lightweight block ciphers against known attacks needs to be analysed.

A new type of attack named invariant subspace attack [136] is of special interest. It was invented in the analysis of a lightweight block cipher PRINTcipher [125]. The design of PRINTcipher is a typical SPN structure with a 3-bit S-box and a Present-like linear layer. It has no key schedule, and the round constants are XORed partially to the state. The flaw in this design is that the round function maps an affine subspace to its coset, which leads to a distinguisher covering any number of rounds for a set of weak keys. The discovery of invariant subspace attack seems ad hoc, but a followed work by Bulygin *et al.* [53] studied the patterns of the invariant subspaces in PRINTcipher and managed to find all subspaces following these patterns. It was left as an open problem whether this attack works for other block ciphers as well, until a generic algorithm to detect the existence of invariant subspaces was proposed by Leander *et al.* [137]. As a result, several lightweight designs have been found to be vulnerable to invariant subspace attacks, such as Zorro [91], iScream [97], and Robin [96]. Notably, the structures of Zorro and the LS-design iScream are different, yet both of them admit a weak-key space in terms of invariant subspace attack. Another victim of the attack is a recently-proposed block cipher Midori [12]; the distinguisher is also found in an ad hoc way, taking advantage of the fixed points in the S-box together with the unfortunate combination of the linear layer and the constants [98]. Unlike differential cryptanalysis [36] and linear cryptanalysis [152] which are extensively studied and comprehensively understood, the critical weakness leading to the invariant subspace attack is not clear. And more importantly, a guideline to preclude such attack needs to be drawn for a "provably secure" framework.

A quick solution to resist the invariant subspace attack is to use heavier key schedules or randomised constants. Indeed, to extend the invariant subspace attack to any number of rounds, it is necessary to make sure that all the round keys and constants are in the weak-key space. A relatively strong key schedule guarantees that the probability for each round key to lie in the weak-key space is negligible. However, in lightweight designs, an ultra-light key schedule reduces the implementation cost. In the meantime, there are designs without key schedule yet having shown no vulnerability to the invariant subspace attack. For instance, no significant weak-key class is found in another LS-design Fantomas [96], and a slight modification of the constant addition

in PRINTcipher will prevent all invariant subspaces of the detected type [53]. A paper by Beierle *et al.* [18] in CRYPTO 2017 considered subspaces which are invariant for both the S-box layer and the linear layer, and focused on the "invariant factors" of the linear layer which provides criteria for resistance against invariant subspace attacks, whereas possible invariant subspaces for the full round function but not the individual components are not considered. Another related research is the subspace trail cryptanalysis proposed by Grassi *et al.* [94], which focuses on the (not necessarily invariant) subspace propagations.

In the rest of this chapter, we denote an $n$-bit vector in $\mathbb{F}_{2^n}$ by $x = (x_{n-1}, x_{n-2}, \ldots, x_0)$. A $k$-dimensional affine subspace in $\mathbb{F}_{2^t}^m$ is denoted by $W = (W_1, W_2, \ldots, W_m)$ where $W_i$ is the subspace of $W$ restricted to a $t$-dimensional subspace of $\mathbb{F}_2^k$. The component function $f_\lambda$ of a vectorial Boolean function $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ is defined as $\lambda \cdot f$, where $\lambda \in \mathbb{F}_{2^n}$ and $\cdot$ is the inner product.

### 4.1.1 Linear properties of a nonlinear function

Nonlinear functions play a critical role in the confusion of cryptographic algorithms. The study of nonlinear functions provides criteria for resistance against various attacks. The S-box of a block cipher is expected to have no linear structures, which might be extended to a linear structure of the whole cipher. One of the early investigations to the existence of linear structures in block ciphers was by Evertse in 1987 [84].

Recall the definition of higher-order differences in Definition 11. Let $F$ be a vectorial boolean function from $\mathbb{F}_{2^n}$ to $\mathbb{F}_{2^m}$. For $a, b \in \mathbb{F}_{2^n}$, the first order derivative is defined as $D_a F(x) = F(x) \oplus F(x \oplus a)$, and the second order derivative is $D^{(2)}_{a,b} F(x) = F(x \oplus a \oplus b) \oplus F(x \oplus a) \oplus F(x \oplus b) \oplus F(x)$.

**Definition 12** ([57, 84] (linear structure, linear kernel))**.** *Let $f$ be a boolean function from $\mathbb{F}_{2^n}$ to $\mathbb{F}_2$. Then, the linear kernel of $f$ includes all vectors $e$ such that $D_e f$ is constant. Any element $e$ of the linear kernel of $f$ is called a linear structure.*

It has been pointed out in [84] that block ciphers with linear structures are vulnerable to attacks much, and there are many works on the characteristics of such structures [78, 84]. However, these properties posed no major threat to any practical ciphers. Two decades later, the study of linear relations between ciphertexts and plaintexts was brought up again in FSE 2013 [50], where a new criterion for avoiding the propagation of linear relations is defined by Boura and Canteaut.

**Definition 13** ([50] ($(v,w)-$linear)). *Let $S$ be a function from $\mathbb{F}_{2^n}$ to $\mathbb{F}_{2^m}$. Then, $S$ is said to be $(v,w)-$linear if there exist two linear subspaces $V \in \mathbb{F}_{2^n}$ and $W \in \mathbb{F}_{2^m}$ with $\dim V = v$ and $\dim W = w$, such that, for all $\lambda \in W$, $S_\lambda$ has degree at most 1 on all cosets of $V$.*

The authors of [50] showed that a function $S$ is $(v,w)-$linear if and only if the component functions $S_\lambda$, $\lambda \in W$ have the property that all its second-order derivatives $D^{(2)}_{\alpha,\beta}S_\lambda$ with $\alpha, \beta \in V$ vanish. Therefore, the definition of $(v,w)-$linear is more general than that of linear structures and linear kernels, which leads to the following proposition.

**Proposition 3.** *Let $S$ be a function from $\mathbb{F}_{2^n}$ to $\mathbb{F}_{2^m}$, $V \in \mathbb{F}_{2^n}$ and $W \in \mathbb{F}_{2^m}$ are subspaces with dimension $v$ and $w$, respectively. If the component functions $S_\lambda$ for all $\lambda \in W$ have a linear kernel $V$, then $S$ is $(v,w)-$linear.*

*Proof.* When the component function $S_\lambda$ has a linear kernel $V$, the first-order derivative of $S_\lambda$ over $V$ is constant for every $\lambda \in W$, hence the second-order derivatives over $V$ always vanish. □

The definition of $(v,w)-$linear property stemmed from the analysis of the hash function Hamsi [50, 88]. Based on the observation that some output bits of the S-box are linearly dependent on certain input bits when the remaining input bits are fixed to constants, such linear relations inside the S-box are extended to the round function and lead to a second pre-image attack to the whole hash function. A similar but more powerful attack [100] was recently proposed on the round-reduced permutation Keccak-$f$ of the SHA-3 hash function. The distinguisher is called a linear structure of Keccak, however in essence, it is closer to the notation of $(v,w)$-linearity.

Next, we will show that the existence of invariant subspaces is closely related to the linear relations inside the S-boxes.

### 4.1.2 Characterise Subspace Propagations in the S-box Layer

Suppose that the round function $F$ is composed with an S-box layer $F_s$, a linear layer $F_l$ and a key addition $F_k$, where $F = F_k \circ F_l \circ F_s$. It is shown in [136] that if there exists an affine subspace $v + A$ that is stable under $F$, $F(v + A) = v + A$, then when the round key $k \in v + u + A$, we have

$$(F_l \circ F_s)(v + A) = u + A\,. \tag{4.1}$$

This means that the invariant subspace property in the round function of a key-alternating block cipher is equivalent to the propagation of special affine

subspaces, namely an affine subspace $v + A$ is transformed to its coset $u + A$. Since the inverse of a linear layer is also linear, Equation (4.1) is equivalent to $F_s(v + A) = F_l^{-1}(u) + F_l^{-1}(A)$, where the right hand side is still an affine subspace. Therefore, we will focus on the propagation of affine subspaces through a layer of S-boxes.

**Definition 14.** *Let $f$ be a (nonlinear) function from $\mathbb{F}_{2^n}$ to $\mathbb{F}_{2^m}$. If an affine subspace $(v + A) \subset \mathbb{F}_{2^n}$ is mapped to $(u + B) \subset \mathbb{F}_{2^m}$ which is also an affine subspace, then $(v + A \to u + B)$ is called an affine subspace propagation.*

Notice that a similar definition is proposed in [94] where subspace trails are studied to construct new impossible differential distinguishers. Here we show that the number of affine subspace propagation is an affine-invariant property.

**Proposition 4.** *The number of affine subspace propagations of vectorial Boolean functions is affine invariant.*

*Proof.* Let $f$ and $g$ be two affine-equivalent vectorial Boolean functions where $g(x) = f(Px)$ with $P$ be an affine transformation. If $(v + A \to u + B)$ is an affine subspace propagation in $f$, $g(P^{-1}(v + A)) = f(v + A) = u + B$, so $(P^{-1}(v+A) \to u+B)$ is an affine subspace propagation of $g$. Hence the number of affine subspace propagations is the same for two affine-equivalent S-boxes. □

In Section 4.1.1, we have shown that if an S-box is $(v, w)-$linear, the corresponding component function is of degree at most 1 over all cosets of $V$. However, in most applications, the property only holds for certain choices of constants, *i.e.*, not for all cosets. Therefore, we introduce the following notion to generalise the property of $(v, w)-$linearity.

**Definition 15.** *Let $f$ be a function from $\mathbb{F}_{2^n}$ to $\mathbb{F}_{2^m}$. Then, $f$ is called linear with respect to $(V, W)$ if there exist two affine subspaces $V \in \mathbb{F}_{2^n}$ and $W \in \mathbb{F}_{2^m}$ with $\dim V = v$ and $\dim W = w$, such that, for all $\lambda \in W$, $f_\lambda$ has degree at most 1 on $V$.*

**Example 2.** *Let the algebraic normal form of an S-box $S : (x_3, x_2, x_1, x_0) \to (y_3, y_2, y_1, y_0)$ be*

$$\begin{aligned}
y_0 &= x_1x_2 + x_0 + x_2 + x_3 \\
y_1 &= x_0x_1x_2 + x_0x_1x_3 + x_0x_2x_3 + x_2x_3 + x_1x_3 + x_3 + x_1 \\
y_2 &= x_0x_1x_3 + x_0x_2x_3 + x_1x_3 + x_0x_3 + x_0x_1 + x_3 + x_2 + 1 \\
y_3 &= x_0x_2x_3 + x_0x_1x_3 + x_0x_1x_2 + x_1x_2 + x_3 + x_1 + x_0 + 1
\end{aligned} \tag{4.2}$$

*A 2-dimensional subspace* $\{0,1,4,5\}$ *is mapped to* $\{c,5,9,0\}$ *of dimension 2, which is equivalent to* $S(0,x_2,0,x_0) = (x_0+1, x_2+1, 0, x_0+x_2)$*, where* $x_2, x_0 \in \mathbb{F}_2$*. However, the image of its coset* $\{8,9,c,d\}$ *is* $\{3,e,4,7\}$*, which is not a 2-dimensional affine subspace.*

**Theorem 2.** *Let* $S$ *be an* $s$*-bit S-box. Then the image of a 2-dimensional affine subspace* $v+A$ *under the S-box is also an affine subspace* $u+B$ *of dimension 2 if and only if the S-box is linear with respect to* $(v+A, \mathbb{F}_{2^n})$*.*

*Proof.* ($\Rightarrow$) When the outputs form an affine subspace of dimension 2, we have that

$$\bigoplus_{x \in v+A} S(x) = 0\,, \tag{4.3}$$

which means that the ANF of the S-box is of degree at most 1 over $v+A$. Indeed, since $v+A$ is of dimension 2, we assume that $x_i, x_j$ are the variables and all the rest are fixed to some constants. Once there exists a term $x_i x_j$ in the ANF of the component function, Equation (4.3) does not always evaluate to zero, which leads to a contradiction.

($\Leftarrow$) The linearity of $S$ with respect to $(v+A, \mathbb{F}_{2^n})$ is equivalent to that the restriction $S|_{v+A}$ is linear; therefore, the image of $v+A$ is also an affine subspace. □

In most lightweight block cipher designs, optimal 3-bit and 4-bit S-boxes are selected to obtain optimised performance in hardware. Meanwhile, some lightweight designs are also the target of known invariant subspace attacks. Therefore, in the following, we will limit the scope of our study to optimal 3-bit and 4-bit S-boxes. Note that a similar result was found in the analysis of invariant subspaces in MIDORI64 [98].

It is known that all 4-bit optimal S-boxes are in one of the 16 affine-equivalent classes [69, 138]. Since the propagation of affine subspaces is affine invariant, we check the representatives in each class of the optimal 4-bit S-boxes. It is shown that no optimal 4-bit S-box admits 3-dimensional affine subspace propagations. However many transitions of 2-dimensional and 1-dimensional affine subspaces are found. Similarly, there is no propagation of 2-dimensional affine subspace in the optimal 3-bit S-boxes. The following theorem characterises the linear dependence of outputs and inputs through an optimal 3-bit and 4-bit S-box.

**Theorem 3.** *Let* $S$ *be an optimal* $s$*-bit S-box with* $s = 3, 4$*. Then the image of an affine subspace* $v+A$ *with dimension less than* $s$ *under the S-box is also an affine subspace* $u+B$ *if and only if* $S$ *is linear with respect to* $(v+A, \mathbb{F}_{2^s})$*.*

*Proof.* When the transition is between two affine subspaces of dimension 1, every bit of the output is a linear function of the input, therefore the conclusion always holds as a trivial case. Since there is no propagation of affine subspaces with dimension 2 for the optimal 3-bit S-boxes, if $S$ maps an affine subspace $v + A$ to an affine subspace $u + B$, then $S$ is linear over $v + A$.

For 4-bit S-boxes, the affine subspace propagations are only of dimension 1 and 2, so the conclusion follows from Theorem 2. □

The theorem shows that if there is a propagation of affine subspaces through an S-box, the S-box acts as a linear function over the input space, *i.e.*, $S$ restricted to the input affine subspace $S|_{v+A}$ is linear. As a special case, if the input subspace is $\mathbb{F}_2^s$, the output is also $\mathbb{F}_2^s$. The conclusion can be extended to a layer of S-boxes.

**Theorem 4.** *Let $F = (S_0, S_1, \ldots, S_{b-1})$ be a layer of optimal $s$-bit ($s = 3, 4$) S-boxes. Then there exists an affine subspace $v+A = (v_0+A_0, v_1+A_1, \ldots, v_{b-1}+A_{b-1})$ whose image through the S-box layer is also an affine subspace $u + B = (u_0 + B_0, u_1 + B_1, \ldots, u_{b-1} + B_{b-1})$ if and only if $v_i + A_i = \mathbb{F}_2^s$ or $F$ restricted to $v_i + A_i$ is a linear transformation, $0 \le i \le b - 1$.*

*Proof.* For a layer of S-boxes $F = (S_0, S_1, \ldots, S_{b-1})$, consider the restriction of $F$ to each of the S-boxes $S_i$. The affine subspace propagation through $F$ implies that the restriction to each S-box also admits a propagation of affine subspaces. Hence the conclusion can be drawn by applying Theorem 3 to each restriction. □

**Example 3.** *We take the affine subspace propagation in* PRINTCIPHER *as an example. The notation follows the original paper, a star $*$ represents the variable taking values in $\mathbb{F}_2$. We refer to [53] for more details. The affine subspace $(1, *, 0)$ is mapped to $(1, *, 1)$ by the S-box $S = \{0, 1, 3, 6, 7, 4, 5, 2\}$ when $k_2$ is 00. Hence the S-box restricted on $(1, *, 0)$ is linear which has a matrix form*

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}. \tag{4.4}$$

## 4.2 Bounding the Invariant Subspaces

### 4.2.1 AES-like* Ciphers

The success of the AES has inspired many block cipher designs with a similar structure. The state can be arranged as a $4 \times 4$ matrix as shown below, with each element of the state consisting of 4-bit or 8-bit.

$$\begin{bmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{bmatrix}$$

The round function of an AES-like cipher includes SubBytes (or SubCells), ShiftRows (or ShuffleCells), MixColumns, KeyAdd and ConstAdd. The first three operations are on 4-bit or 8-bit words, which means there are no bit-level operations. Here we focus on the lightweight AES-like ciphers with 4-bit S-boxes, and denote them by AES-like*; they include the following operations.
SubCells: The 4-bit S-box is applied to each of the 16 cells.
ShuffleCells: A shuffle on the 16 cells of the state. In most of the designs, the ShuffleCells operation shuffles the cell in a way that the cells belonging to the same column are distributed over 4 different columns to achieve good diffusion.
MixColumns: A linear transformation $M$ operates on the 4 columns of the state. Normally $M$ is selected from MDS or near-MDS matrices.
KeyAdd&ConstAdd: The key schedule and constants are usually extremely lightweight.

MIDORI is a lightweight block cipher for low energy devices. It has two instances MIDORI64 and MIDORI128, with 64-bit and 128-bit block size, respectively. MIDORI64 is a typical AES-like* cipher. Its round function contains the following operations
SubCells: The 4-bit S-box $S_0[16] = \{\mathtt{C}, \mathtt{A}, \mathtt{D}, \mathtt{3}, \mathtt{E}, \mathtt{B}, \mathtt{F}, \mathtt{7}, \mathtt{8}, \mathtt{9}, \mathtt{1}, \mathtt{5}, \mathtt{0}, \mathtt{2}, \mathtt{4}, \mathtt{6}\}$.
ShuffleCells: Each cell of the state is shuffled:
$(s_0, s_1, ..., s_{15}) \leftarrow (s_0, s_{10}, s_5, s_{15}, s_{14}, s_4, s_{11}, s_1, s_9, s_3, s_{12}, s_6, s_7, s_{13}, s_2, s_8)$.
MixColumns: The matrix $M$ is applied to each column of the state.

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}. \tag{4.5}$$

KeyAdd&ConstAdd: The round key is derived from the master key, and the round constants are in $\{0,1\}^{16}$. We refer to [12] for the details of MIDORI.

### 4.2.2 Bounds on Invariant Subspace Attacks in AES-like*

The invariant subspace attacks have been found until now in two ways; through ad hoc or heuristic search. Rather than checking possible attacks after every adjustment of parameters and components, it is preferable for designers to have a guideline to preclude the existence of large invariant subspaces during the design process. Based on the discussions in the previous section, we will show that the dimension of some invariant subspaces in AES-like* ciphers can be bounded.

Assume that there exists an invariant subspace $A = (A_1, A_2, \ldots, A_{16})$ in the round function of an AES-like* block cipher. We ignore the operations KeyAdd and ConstAdd for a moment, since they have no influence on the dimension of the affine subspaces. According to Theorem 4, the output sets after SubCells, ShuffleCells are also affine subspaces. Hence we denote the output sets $B, D, E$ after SubCells, ShuffleCells and MixColumns by $B = (B_1, B_2, \ldots, B_{16})$, $D = (D_1, D_2, \ldots, D_{16})$ and $E = (E_1, E_2, \ldots, E_{16})$, as illustrated below.

$$\begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_{15} \\ A_{16} \end{bmatrix} \xrightarrow{\text{SubCells}} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_{15} \\ B_{16} \end{bmatrix} \xrightarrow{\text{ShuffleCells}} \begin{bmatrix} D_1 \\ D_2 \\ \vdots \\ D_{15} \\ D_{16} \end{bmatrix} \xrightarrow{\text{MixColumns}} \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_{15} \\ E_{16} \end{bmatrix}.$$

We further assume that the input space $A = (A_1, A_2, \ldots, A_{16})$ has the property that the restrictions $A_i$ over each S-box are linearly independent from each other. Then, we have the following bound on the dimension of the invariant subspaces when the MixColumns matrix is MDS.

**Theorem 5.** *Let $f$ be the round function without key and constant addition of an AES-like* cipher with an MDS MixColumn layer, satisfying that the ShuffleCells operation spreads cells from the same column to all different columns. When the input space $A = (A_1, A_2, \ldots, A_{16})$ is such that the restrictions $A_i$ over each S-box are linearly independent from each other, the dimension of any invariant subspace is at most 32.*

*Proof.* We show that the restriction of the invariant subspace $A$ over every S-box is of dimension at most 2. Firstly, the subspaces propagating through 4-bit S-boxes are of dimension 1, 2, or 4, and the dimensions of the intermediate outputs through the S-box layer remain the same, hence $\sum \dim(A_i) = \sum \dim(B_i)$. Note here that the sum of the dimensions of the restricted spaces only equals the dimension of the input space when the restrictions are linearly independent

from each other. While the ShuffleCells operation simply permutes the cells, the only operation that has an influence on the dimensions of the restricted subspaces is the MixColumns layer. For the four cells involved in a MixColumns operation, w.l.o.g $(E_1, E_2, E_3, E_4)$, we consider the following cases.
**Case 1.** Suppose that there is one cell taking values in $\mathbb{F}_2^4$. Since all the entries in an MDS matrix are nonzero (the inverse of an MDS matrix is also MDS), the dimensions of $D_i, i = 1, 2, 3, 4$ are also 4.
**Case 2.** Suppose that there are two or three cells taking all values in $\mathbb{F}_2^4$. Since all the entries of the MDS matrix are nonzero, and the values in each cell are independent from other cells, the dimensions of $D_i, i = 1, 2, 3, 4$ are also 4.

So for **Case 1** and **2**, $\sum \dim(D_i) > \sum \dim(E_i) = \sum \dim(A_i)$. However, we know from the SubCells layer that $\sum \dim(A_i) = \sum \dim(B_i)$, thus $\sum \dim(D_i) > \sum \dim(B_i)$, which contradicts the fact that the ShuffleCells operation only permute the cells without changing their dimensions.
**Case 3.** Suppose that all four cells take values in $\mathbb{F}_2^4$, then the dimensions of $D_i, i = 1, 2, 3, 4$ are also 4. Recall that in the construction of the ShuffleCells, in order to achieve good diffusion, it is preferably to spread the cells from the same column to different columns. That is to say the cells of dimension 4 are distributed to 4 different columns. As a result, Case 3 either yields a contradiction with Case 1 or a trivial case where all 16 cells receive inputs in $\mathbb{F}_2^4$.

To conclude, if there exists an invariant subspace in an AES-like* cipher with an MDS MixColumns layer, every cell can only take a value from subspaces of dimension at most 2, which means the total dimension is at most 32. □

The main observation of the above theorem is that the restriction of an invariant subspaces on any cell cannot be of dimension 4, when the MixColumns matrix is MDS. The observation can be generalised for any matrix by verifying a simple condition, as we will discuss next.

**Definition 16.** *For a subspace $W = (W_1, W_2, \ldots, W_m) \in \mathbb{F}_{2^s} \times \mathbb{F}_{2^s} \times \ldots \mathbb{F}_{2^s}$, we define an $m$-bit vector $AI(W) = w = (w_1, w_2, \ldots, w_m)$ as the active indicator of $W$, where*

$$w_i = \begin{cases} 1, W_i = \mathbb{F}_{2^n}; \\ 0, \textit{otherwise.} \end{cases}$$

*It is obvious that the nonzero entries of $w$ indicates the active cells in $W$.*

**Definition 17.** *Let $P$ be an $n \times n$ matrix over $\mathbb{F}_{2^s}$, $u$ be a binary vector of length $n$. We define $P \odot u$ by a binary vector $v$ of length $n$, where $v_i = \bigvee_j P_{i,j} \cdot u_j$ and $\bigvee$ is the OR operation.*

**Theorem 6.** *Suppose $M : \mathbb{F}_{2^s}^m \to \mathbb{F}_{2^s}^m$ is linear. Let $W = (W_1, W_2, \ldots, W_m)$ be a subspace and $V = (V_1, V_2, \ldots, V_m)$ be the image of $W$ under the linear function $M$. Then, $W$ has the same active pattern as $V$ if $M \odot AI(W) = AI(W)$.*

*Proof.* The sets in each $V_i$ are a linear combination of $W_j$, $1 \leq j \leq m$. If there exists a $j_0$ such that $M_{i,j_0} \cdot AI(W)_{j_0} = 1$, then $AI(V)_i = 1$. Thus, $W$ has the same active pattern with $V$ if $\bigvee_j M_{i,j} \cdot AI(W)_j = AI(W)_i$, *i.e.* $M \odot AI(W) = AI(W)$. □

It is obvious that the trivial cases are $AI(W) = (0, 0, \ldots, 0)$ or $(1, 1, \ldots, 1)$. If $(0, 0, \ldots, 0)$ and $(1, 1, \ldots, 1)$ are the only solutions to the equation $M \odot x = x$, it means that if there exists an invariant subspace for a round function with the linear layer $M$, its dimension is at most 32, which corresponds to $AI(W) = (0, 0, \ldots, 0)$ ($AI(W) = (1, 1, \ldots, 1)$ is the whole space).

There are AES-like* lightweight proposals with non-MDS MixColumns layer such as MIDORI64 and SKINNY64 [19]. The linear function $M$ is the composition of ShuffleCells and MixColumns, which is a $16 \times 16$ matrix over $\mathbb{F}_{2^4}$. Here we take the round function of MIDORI64 without key and constant addition as an example and assume that the input cells are linearly independent. The linear layer has a matrix form as given by Equation (4.6) below.

$$\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0
\end{pmatrix} \tag{4.6}$$

It can be verified that $M \odot x = x$ only has trivial solutions. Therefore, for the round function of MIDORI64 without the addition of keys and constants, if there are any invariant subspaces with independent input cells, each cell is of dimension at most 2. Recall that the round constants of MIDORI for each cell

are 0 or 1, hence, they are more likely to fall to some aforementioned invariant subspaces. It can also be seen as a general explanation of the "unfortunate combination of constants" as described in the discovery of the distinguisher [98]. The same upper bound on the dimension can be deduced for SKINNY64; the details of the deduction is omitted here.

### 4.2.3 Countermeasures and Discussions

Countermeasures to resist the invariant subspace attack can be deduced based on Theorem 5 and 6. Firstly we study the properties of the invariant subspaces in the keyless and constant-less round function of a given cipher, based on which a guideline for the choice of the constants can be deduced. For the round function of AES-like* ciphers with MDS MixColumns layer and MIDORI64, when considering the invariant subspaces with independent inputs on each cells, each cell should be of dimension at most 2.

As a consequence, we deduce a countermeasure as follows. Denote the set of the constants for a single cell over all rounds by $C_{const}$. The designer chooses $C_{const}$ such that there is no 2-dimensional subspace $V$ of $\mathbb{F}_2^4$ satisfying

$$C_{const} \subseteq V.$$

For instance, if the constants for the first cell $s_0$ are $\{1, 2, 3, 4, 5\}$ over the first five rounds, then clearly $\{1, 2, 3, 4, 5\}$ can not fall to any 2-dimensional subspace, hence the propagation of any nontrivial invariant subspaces with independent inputs on each cell will be blocked after five rounds. Hence, invariant subspace attack cannot be found for arbitrary number of rounds. It is obvious that this is much more lightweight, compared with a heavier key schedule or complex constants.

**Remark 3.** *The bound and countermeasures are tailored for AES-like* primitives, which are suitable for the majority of current lightweight designs. While the bounding technique does not directly apply to bit-level operations, we leave the generalisation to bit-based designs as an open problem.*

## 4.3 Conclusions

In this chapter, we focused on finding a mathematical characteration of the (non)existence of invariant subspaces in lightweight block ciphers. By studying the propagation of subspaces in S-boxes, we deduce that if $v + A$ is an invariant subspace in the round function of a lightweight block cipher with 3-bit or

4-bit S-box, then $v + A$ restricted to each S-box has dimension 4 or at most 2. Furthermore, when $v + A$ restricted to $S_i$ is of dimension at most 2, $S_i$ is linear over $v + A|_{S_i}$. As a consequence, we focus on lightweight AES-like ciphers with 4-bit S-boxes and an MDS MixColumn layer. The dimension of an invariant subspace is upper bounded by 32, and the round function restricted to the invariant subspace is linear. Based on the results, we propose a countermeasure to resist invariant-subspace distinguishers in some lightweight ciphers, which is easy and lightweight.

# Chapter 5

# Nonlinear Diffusion Layers

## 5.1 Motivation

Confusion and diffusion are the main criteria for cryptographic building blocks, especially in the design of block ciphers. A prominent type is the SPN ciphers, where confusion and diffusion are achieved mainly by the S-boxes and the permutation layer, respectively. The common use of the SPN structures can partially be attributed to the security proof it inherits from the structure, as in the Wide Trail Strategy. In this framework, the major feature of a permutation layer is its branch number, which is related to the minimum distance of an error-correcting code. For instance, the diffusion layer of the AES is based on a linear MDS code with length 8, dimension 4 and distance 5 over $\mathbb{F}_{2^8}$. Most AES-like ciphers have a diffusion layer based on a linear code. In fact, in the practice of block cipher design, a consensus seems to have developed about the diffusion function: designers choose linear functions with large branch numbers to achieve provable bounds against differential and linear cryptanalysis.

Despite the clear division of labour for the S-boxes and the linear layer in SPN ciphers, the Wide Trail Strategy does not require that the diffusion layer is linear, see for example [172, p.142]. Intuitively speaking, a nonlinear diffusion layer is able to guarantee a similar provable bound compared to a linear one with the same branch number, while the confusion of the cipher may benefit as well. However, no alternative to linear diffusion layers has been proposed or studied in the literature in the context of the Wide Trail Strategy. A possible reason is that the properties of nonlinear error-correcting codes are less explored than the linear ones, which sets limit to the design and the analysis.

In this chapter, we present several constructions of nonlinear diffusion layers. One is based on a well-known nonlinear code, a member of the Kerdock code family, which is of length 16, 256 codewords and minimal distance 6 over $\mathbb{F}_2$. The other one is a generic construction of nonlinear functions with T-functions, which is of theoretical interests and able to generate a number of good nonlinear diffusion functions, such as the nonlinear functions based on modular additions designed in this chapter. We study the cryptographic properties of the nonlinear diffusion layers, such as their branch number and differential probabilities. From a practical point of view, we also construct example ciphers to illustrate the advantage of nonlinear diffusion layer in the design of lightweight block ciphers.

The rest of this chapter is organised as follows: Section 5.2 recalls the general definition of branch number. Section 5.3 introduces the Kerdock nonlinear code family, derives a nonlinear diffusion function based on an instance $\mathcal{K}(4)$ and analyses its cryptographic properties. We propose a generic construction of nonlinear functions in Section 5.4 as well as a rigorous deduction of their branch numbers. In particular, two instances are designed based on modular additions which possess a diffusion property compatible with an NMDS function. To illustrate the advantage of the nonlinear diffusion functions, we show two example ciphers in Section 5.5, and analyse the minimum number of active S-boxes and the expected differential probability for the differential characteristics. Section 5.6 concludes the chapter.

## 5.2 A General Definition of Branch Number

Let $n, t$ be positive integers. The value of $a$ in Hexadecimal is shown with numbers in `typewriter` font. An element in the vector space over a finite field $\mathbb{F}_{2^t}^n$ is represented as a vector $\hat{x} = \{x_{n-1}, \cdots, x_1, x_0\}$, while a vector of length $n$ over $\mathbb{Z}_{2^t}$ the ring of integers modulo $2^t$ is denoted with bold letters, *i.e.*, $\hat{\boldsymbol{y}} = \{\boldsymbol{y_{n-1}}, \cdots, \boldsymbol{y_1}, \boldsymbol{y_0}\}$. The $j$-th bit of the $i$-th component of an element $a \in \mathbb{F}_{2^t}$ is denoted by $a_i[j]$. We denote matrices of dimension $n \times n$ over a finite field with capital letters, and $n \times n$ matrices over a ring with bold font.

Diffusion layers are one of the core components in block ciphers. One of the metrics to measure a good diffusion function is the branch number which is often defined for linear functions over finite fields. However, it is possible to define them for general functions.

**Definition 18.** *Let $(G, \otimes)$ be an Abelian group, $n$ be an integer. The differential branch number of a (possibly nonlinear) function $f : G^n \to G^n$ is defined as*

$$B_d(f) = \min_{\alpha, \beta \neq \alpha} \{wt(\alpha \otimes \beta^{-1}) + wt(f(\alpha) \otimes f(\beta)^{-1})\}, \alpha, \beta \in G^n,$$

*where $+$ is integer addition, $g^{-1}$ is the inverse element of $g \in G$ and $wt(\cdot)$ is the Hamming weight with $wt(x) = wt(x_{n-1}, \cdots, x_1, x_0) = \#\{i | x_i \neq 0\}$.*

Similarly, linear branch number can be defined as follows.

**Definition 19** ([172])**.** *Let $(G, \otimes)$ be an Abelian group, $n$ be an integer. The linear branch number of a function $f : G^n \to G^n$ is defined as*

$$B_l(f) = \min_{\alpha,\beta,C(\alpha,\beta)\neq 0} \{wt(\alpha) + wt(\beta)\},$$

*where $C(\cdot, \cdot)$ is the correlation with input mask $\alpha$ and output mask $\beta$.*

## 5.3 A Nonlinear Function Based on the Kerdock Code

### 5.3.1 Kerdock Codes

Although much less is known about nonlinear error-correcting codes than about linear codes, a few examples with very good properties are known. Kerdock codes [116, 166] are an example. In order to create a diffusion map similar to MixColumns from a nonlinear code, we put some extra requirements. Firstly, for practical reasons the code should be over an alphabet with size a power of two. Secondly, recall that in AES the action of MixColumns on a single column can be described by $\hat{y} = M\hat{x}$ where $M$ is a matrix derived from a generator matrix $\begin{bmatrix} I & M \end{bmatrix}$ of the corresponding linear code, where $\hat{x}, \hat{y}$ are vectors of length 4 over $\mathbb{F}_{2^8}$. In other words, the matrix $M$ is the matrix used to compute the redundancy symbols of the codeword (= output of MixColumns) from the data symbols (= input of MixColumns). Hence, if we use a nonlinear code to define a diffusion map in a similar way, then there should be an explicit formula to compute the redundancy symbols from the data symbols. This means that we need a *systematic* code. Furthermore, we require that the binary logarithm of the number of codewords equals the length of the codewords, divided by two.

We now explain Kerdock codes following the derivation of [143]. Consider the space $V$ of all bit-strings of length $m$ and let $f$ be a function from $V$ to $\{0, 1\}$. Define a codeword $\hat{c}_f$ with length $n = 2^m$ as follows:

$$\hat{c}_f = (f(00\ldots0), f(00\ldots1), \ldots, f(11\ldots1)) .$$

Then the first-order Reed-Muller code of length $n = 2^m$ is defined as follows:

$$\mathcal{R}(1, m) = \{\hat{c}_f \mid f \text{ is a linear/affine function on } V\} .$$

Since the sum of two linear/affine functions is a linear/affine function, it follows that $\mathcal{R}(1,m)$ is a linear code. Since there are $2^{m+1}$ linear/affine functions over $V$, $\mathcal{R}(1,m)$ has dimension $m+1$. The second-order Reed-Muller code of length $n=2^m$ is defined as:

$$\mathcal{R}(2,m) = \{\hat{c}_f \mid f \text{ is a function of degree at most 2 on } V\}\,.$$

Also $\mathcal{R}(2,m)$ is a linear code. $\mathcal{R}(2,m)$ can also be described as

$$\begin{aligned}\mathcal{R}(1,m) = \{\hat{c}_{f+g} \mid f \text{ is a linear/affine function on } V \text{ and} \\ g \text{ is a homogeneous function of degree 2 on } V\}\,.\end{aligned} \tag{5.1}$$

The Kerdock code of length $m$ satisfies $\mathcal{R}(1,m) \subset C \subset \mathcal{R}(2,m)$. In fact Kerdock codes can be formed by putting extra conditions on $g$ in Equation (5.1). A homogeneous function of degree 2 can be described by a quadratic form:

$$Q(\hat{x}) = \sum_{1\le i<j\le m} x_i x_j\,, \tag{5.2}$$

or the corresponding *symplectic* form:

$$B(\hat{x},\hat{y}) = Q(\hat{x}+\hat{y}) - Q(\hat{x}) - Q(\hat{y}) = \hat{x}B\hat{y}^T\,, \tag{5.3}$$

where $B$ is a *symplectic* matrix ($B=-B^T$ and all elements on the main diagonal are equal to 0). A set of $2^{m-1}$ symplectic $(m\times m)$-matrices ($m$ even) such that the difference of any two matrices of the set is nonsingular is called a *Kerdock set*.

Let $m$ be even and $m\ge 4$. Let $G$ be the set of homogeneous functions of degree 2 defined by the $2^{m-1}$ matrices of a Kerdock set according to Equations (5.2)–(5.3). Then the *Kerdock code* of length $n=2^m$ is defined as follows:

$$\mathcal{K}(m) = \{\hat{c}_{f+g} \mid f \text{ is a linear/affine function on } V \text{ and } g\in G\}. \tag{5.4}$$

Since there are $2^{m+1}$ choices for $f$ and $2^{m-1}$ choices for $g$, we obtain that $\mathcal{K}(m)$ contains $2^{2m}$ codewords. It can be shown that the minimum distance of $\mathcal{K}(m)$ equals $2^{m-1}-2^{m/2-1}$. In addition, all Kerdock codes are systematic.

When $m=4$, $\mathcal{K}(4)$ has length $2^4=16$ and contains $2^{2\cdot 4}=2^8$ codewords. It has minimum distance $2^3-2^{2-1}=6$. This code is also known as the Nordstrom-Robinson code [166]. Table 5.1 illustrates that Kerdock codes for larger $m$ do not satisfy the requirements that we put at the start of this section.

Table 5.1: Parameters of Kerdock codes for $4 \leq m \leq 8$

| $m$ | $n = 2^m$ | $\log_2(\lvert C \rvert) = 2m$ | $d = 2^{m-1} - 2^{m/2-1}$ |
|---|---|---|---|
| 4 | 16 | 8 | 6 |
| 6 | 64 | 12 | 28 |
| 8 | 256 | 16 | 120 |

Table 5.2: Lookup table of the nonlinear function $\zeta$

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 76 | B9 | CF | 97 | 2D | E2 | 58 | 6B | D1 | 1E | A4 | FC | 8A | 45 | 33 |
| 1 | E5 | 9C | 53 | 2A | 4E | FB | 34 | 81 | B2 | 7 | C8 | 7D | 19 | 60 | AF | D6 |
| 2 | DA | A3 | 6C | 15 | 71 | C4 | B | BE | 8D | 38 | F7 | 42 | 26 | 5F | 90 | E9 |
| 3 | 3F | 49 | 86 | F0 | A8 | 12 | DD | 67 | 54 | EE | 21 | 9B | C3 | B5 | 7A | C |
| 4 | AE | 1B | D4 | 61 | C9 | B0 | 7F | 6 | 35 | 4C | 83 | FA | 52 | E7 | 28 | 9D |
| 5 | 78 | C2 | D | B7 | 23 | 55 | 9A | EC | DF | A9 | 66 | 10 | 84 | 3E | F1 | 4B |
| 6 | 47 | FD | 32 | 88 | 1C | 6A | A5 | D3 | E0 | 96 | 59 | 2F | BB | 1 | CE | 74 |
| 7 | 91 | 24 | EB | 5E | F6 | 8F | 40 | 39 | A | 73 | BC | C5 | 6D | D8 | 17 | A2 |
| 8 | 5D | E8 | 27 | 92 | 3A | 43 | 8C | F5 | C6 | BF | 70 | 9 | A1 | 14 | DB | 6E |
| 9 | 8B | 31 | FE | 44 | D0 | A6 | 69 | 1F | 2C | 5A | 95 | E3 | 77 | CD | 2 | B8 |
| A | B4 | E | C1 | 7B | EF | 99 | 56 | 20 | 13 | 65 | AA | DC | 48 | F2 | 3D | 87 |
| B | 62 | D7 | 18 | AD | 5 | 7C | B3 | CA | F9 | 80 | 4F | 36 | 9E | 2B | E4 | 51 |
| C | F3 | 85 | 4A | 3C | 64 | DE | 11 | AB | 98 | 22 | ED | 57 | F | 79 | B6 | C0 |
| D | 16 | 6F | A0 | D9 | BD | 8 | C7 | 72 | 41 | F4 | 3B | 8E | EA | 93 | 5C | 25 |
| E | 29 | 50 | 9F | E6 | 82 | 37 | F8 | 4D | 7E | CB | 4 | B1 | D5 | AC | 63 | 1A |
| F | CC | BA | 75 | 3 | 5B | E1 | 2E | 94 | A7 | 1D | D2 | 68 | 30 | 46 | 89 | FF |

### 5.3.2 Diffusion Function $\zeta$ Based on $\mathcal{K}(4)$

**Definition 20.** *We define the diffusion function $\zeta$ by stating that, for all 8-bit strings $x$, the image $\zeta(x)$ is the unique 8-bit value $y$ such that the 16-bit value $x \| y$ is a codeword of $\mathcal{K}(4)$.*

Table 5.2 gives a tabular representation of $\zeta$.

#### Diffusion Properties of $\zeta$

**Theorem 7.** *The nonlinear function $\zeta$ has differential branch number* 6.

*Proof.* By definition, the differential branch number of the nonlinear function is the minimum distance between the codewords of $\mathcal{K}(4)$, which equals 6. □

The advantage of the function $\zeta$ is two-fold. The minimum distance of a binary linear code with length 16 and dimension 8 is at most 5 [195], which means $\zeta$ achieves better diffusion than any linear layer operating on $\mathbb{F}_2^8$. In addition, the nonlinearity of $\zeta$ may lower the maximum expected differential probability of the optimal characteristics, when $\zeta$ is adopted in stead of the linear layer of a cipher. This depends of course on the cipher.

### Differential Property of $\zeta$

An alternative description of the Kerdock code $\mathcal{K}(4)$ is the binary image of a $\mathbb{Z}_4$-linear code with the generator matrix

$$\boldsymbol{M} = \begin{pmatrix} 3 & 2 & 3 & 1 \\ 3 & 3 & 1 & 2 \\ 2 & 1 & 1 & 1 \\ 3 & 1 & 2 & 3 \end{pmatrix},$$

under the Gray map $\Phi : \mathbb{Z}_4 \to \mathbb{F}_{2^2}$ which is defined by:

$$\Phi(0) = 00, \Phi(1) = 01, \Phi(2) = 11, \Phi(3) = 10.$$

For more information on $\mathbb{Z}_4$-linear codes and their links with nonlinear codes over $\mathbb{F}_2$, we refer to [195]. We know from Section 5.3.1 that the diffusion function $\zeta$ based on $\mathcal{K}(4)$ is quadratic. As a result, its difference distribution table (DDT) has several interesting properties, which reflects the symmetry in the Kerdock codes.

Before diving into the properties, we introduce a few lemmata on the interactions among the Gray map, XOR and modular addition. Their correctness can be verified by going over all the cases.

**Lemma 5.** *Let $\boldsymbol{v_1}, \boldsymbol{v_2} \in \mathbb{Z}_4$. Let $\Phi$ be the Gray map. We have*

$$\begin{aligned} \Phi(\boldsymbol{v_1} \boxplus \boldsymbol{v_2}) &= \Phi(\boldsymbol{v_1}) \oplus \Phi(\boldsymbol{v_2}) \oplus e\,, \\ \Phi(\boldsymbol{v_1} \boxplus \boldsymbol{v_2} \boxplus 2) &= \Phi(\boldsymbol{v_1}) \oplus \Phi(\boldsymbol{v_2}) \oplus \Phi(2) \oplus e\,, \end{aligned} \tag{5.5}$$

*where*

$$e = \begin{cases} 00, & \textit{when } \boldsymbol{v_1} \in \{0, 2\} \textit{ or } \boldsymbol{v_2} \in \{0, 2\} \\ 11, & \textit{otherwise.} \end{cases},$$

$$= \Phi(2\boldsymbol{v_1}\boldsymbol{v_2})\,.$$

**Lemma 6.** *Let $y_1, y_2 \in \mathbb{F}_{2^2}$. $\Phi^{-1}$ is the inverse Gray map. We have*

$$\Phi^{-1}(y_1 \oplus y_2) = \Phi^{-1}(y_1) \boxplus \Phi^{-1}(y_2) \boxplus \boldsymbol{f}\,, \tag{5.6}$$

*where*

$$\boldsymbol{f} = \begin{cases} 0, & \text{when } y_1 \in \{00, 11\} \text{ or } y_2 \in \{00, 11\}, \\ 2, & \text{otherwise,} \end{cases}$$

$$= 2 \cdot \Phi^{-1}(y_1) \cdot \Phi^{-1}(y_2)\,,$$

*and the multiplications are over $\mathbb{Z}_4$.*

Lemma 5 and Lemma 6 show that the behaviour of $0, 2$ is different from that of $1, 3$ with respect to the Gray map. Therefore, we define the following subsets of $\mathbb{Z}_4^4$.

**Definition 21.** *The space $\mathbb{Z}_4^4$ can be divided into 16 sets $\Lambda_i$ with $0 \le i \le 15$, where*

$$\Lambda_i = \{ \begin{bmatrix} \boldsymbol{c_3} & \boldsymbol{c_2} & \boldsymbol{c_1} & \boldsymbol{c_0} \end{bmatrix}^T \mid \boldsymbol{c_j} \in \{0, 2\}, \text{ when the } j\text{-th bit of } i \text{ is } 0 \text{ otherwise } \boldsymbol{c_j} \in \{1, 3\} \}\,. \tag{5.7}$$

*We further denote the binary image of $\Lambda_i$ by $\lambda_i$.*

For example:

$$\Lambda_0 = \{ \begin{bmatrix} \boldsymbol{c_3} & \boldsymbol{c_2} & \boldsymbol{c_1} & \boldsymbol{c_0} \end{bmatrix}^T \mid \boldsymbol{c_j} \in \{0, 2\} \}\,,$$

and

$$\lambda_0 = \{\texttt{00,03,0C,0F,30,33,3C,3F,C0,C3,CC,CF,F0,F3,FC,FF}\}\,,$$

where the numbers in typewriter font are 8-bit strings in hexadecimal notations.

**Theorem 8.** *The differential probability of any input difference in $\zeta$ is 0, 1 or 1/4. In particular, when the input difference is in $\lambda_0$ or $\lambda_{15}$, the output difference is the image of the input difference under $\zeta$ and has differential probability 1.*

*Proof.* Let $\hat{a}, \hat{b}$ be the input and output differences which are vectors in $\mathbb{F}_2^8$. We consider the differential equation:

$$\zeta(\hat{x}) \oplus \zeta(\hat{x} \oplus \hat{a}) = \hat{b}\,,$$

which is equivalent to

$$\Phi(M\Phi^{-1}(\hat{x})) \oplus \Phi(M\Phi^{-1}(\hat{x} \oplus \hat{a})) = \hat{b}\,.$$

By Lemma 6, it is equivalent to

$$\begin{aligned} &\Phi(M\Phi^{-1}(\hat{x})) \oplus \Phi(M(\Phi^{-1}(\hat{x}) \boxplus \Phi^{-1}(\hat{a}) \boxplus \hat{\boldsymbol{f}})) = \hat{b}, \\ \Rightarrow\quad &\Phi(M\Phi^{-1}(\hat{x})) \oplus \Phi(M\Phi^{-1}(\hat{x}) \boxplus M\Phi^{-1}(\hat{a}) \boxplus M\hat{\boldsymbol{f}}) = \hat{b}\,, \end{aligned}$$

where $\hat{\boldsymbol{f}} = \begin{bmatrix} \boldsymbol{f_3} & \boldsymbol{f_2} & \boldsymbol{f_1} & \boldsymbol{f_0} \end{bmatrix}^T$ depends on $\hat{x}$ and $\hat{a}$, specifically $\hat{\boldsymbol{f}} \in \Lambda_0$ with

$$\boldsymbol{f_i} = 2\Phi^{-1}(\hat{x})_i \Phi^{-1}(\hat{a})_i\,.$$

With Lemma 5 and $\hat{\boldsymbol{f}} \in \Lambda_0$, the above equation is equivalent to

$$\begin{aligned} &\Phi(M\Phi^{-1}(\hat{x})) \oplus \Phi(M(\Phi^{-1}(\hat{x}))) \oplus \Phi(M\Phi^{-1}(\hat{a})) \oplus \Phi(M\hat{\boldsymbol{f}}) \oplus \hat{e} = \hat{b}\,, \\ \Rightarrow\quad &\Phi(M\Phi^{-1}(\hat{a})) \oplus \Phi(M\hat{\boldsymbol{f}}) \oplus \hat{e} = \hat{b}\,, \end{aligned} \tag{5.8}$$

where

$$\begin{aligned} \hat{e}_i &= \begin{cases} 00, (M\Phi^{-1}(\hat{x}))_i \in \{0,2\} \text{ or } (M\Phi^{-1}(\hat{a}))_i \in \{0,2\}\,, \\ 11, \text{otherwise.} \end{cases} \\ &= \Phi(2(M\Phi^{-1}(\hat{x}))_i (M\Phi^{-1}(\hat{a}))_i)\,. \end{aligned}$$

Therefore, the output difference is $\hat{b} = \Phi(M\Phi^{-1}(\hat{a})) \oplus \Phi(M\hat{\boldsymbol{f}}) \oplus \hat{e}$. Next, we discuss the possible values of $\Phi(M\hat{\boldsymbol{f}}) \oplus \hat{e}$.

**(1)** If the input difference $\hat{a} \in \lambda_0$, *i.e.*, $\Phi^{-1}(\hat{a}) \in \Lambda_0$. Then $\hat{\boldsymbol{f}} = \vec{0}$, and $\hat{e} = \vec{0}$ since $M\Phi^{-1}(\hat{a}) \in \Lambda_0$ (notice that $\Lambda_0$ is stable under $M$, *i.e.*, $M\Lambda_0 = \Lambda_0$). Thus, $\Phi(M\hat{\boldsymbol{f}}) \oplus \hat{e} = 0$, and we have $\Phi(M\Phi^{-1}(\hat{a})) = \hat{b}$.

**(2)** If $\Phi^{-1}(\hat{a}) \in \Lambda_{15}$, we have $\boldsymbol{f_i} = 2\Phi^{-1}(\hat{x})_i$, thus $M\hat{\boldsymbol{f}} = 2M\hat{x}$, where the multiplication is over $\mathbb{Z}_4$. Hence, $(M\hat{\boldsymbol{f}})_i$ equals 0 when $(M\Phi^{-1}(\hat{x}))_i \in \{0,2\}$ and 2 when $(M\Phi^{-1}(\hat{x}))_i \in \{1,3\}$. Therefore, $\Phi(M\hat{\boldsymbol{f}}) = \hat{e}$, and we have $\Phi(M\Phi^{-1}(\hat{a})) = \hat{b}$.

**(3)** Denote $\Phi^{-1}(\hat{x})$ by $\hat{\boldsymbol{x}}$ and $\Phi^{-1}(\hat{a})$ by $\hat{\boldsymbol{a}}$. Denote by $\hat{\boldsymbol{u}} \cdot \hat{\boldsymbol{v}}$ the component-wise product of two vectors $\hat{\boldsymbol{u}}, \hat{\boldsymbol{v}}$. If $\hat{\boldsymbol{a}}$ is not in $\Lambda_0$ or $\Lambda_{15}$, we have

$$\Phi(M\hat{\boldsymbol{f}}) \oplus \hat{e} = \Phi(M(2\hat{\boldsymbol{x}} \cdot \hat{\boldsymbol{a}})) \oplus \Phi(2(M\hat{\boldsymbol{x}}) \cdot (M\hat{\boldsymbol{a}}))\,,$$

which is equivalent to $\Phi(M(2\hat{\boldsymbol{x}} \cdot \hat{\boldsymbol{a}}) \boxplus (2(M\hat{\boldsymbol{x}}) \cdot (M\hat{\boldsymbol{a}})))$ since $2(M\hat{\boldsymbol{x}}) \cdot (M\hat{\boldsymbol{a}}) \in \Lambda_0$. With $\hat{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x_3} & \boldsymbol{x_2} & \boldsymbol{x_1} & \boldsymbol{x_0} \end{bmatrix}^T$ and $\hat{\boldsymbol{a}} = \begin{bmatrix} \boldsymbol{a_3} & \boldsymbol{a_2} & \boldsymbol{a_1} & \boldsymbol{a_0} \end{bmatrix}^T$, we obtain

$$
\begin{aligned}
& M(2\hat{\boldsymbol{x}}\hat{\boldsymbol{a}}) \boxplus (2(M\hat{\boldsymbol{x}}) \cdot (M\hat{\boldsymbol{a}})) \\
= \quad & M \begin{bmatrix} 2\boldsymbol{x_3}\boldsymbol{a_3} \\ 2\boldsymbol{x_2}\boldsymbol{a_2} \\ 2\boldsymbol{x_1}\boldsymbol{a_1} \\ 2\boldsymbol{x_0}\boldsymbol{a_0} \end{bmatrix} \boxplus 2 \begin{bmatrix} (3\boldsymbol{x_3} \boxplus 2\boldsymbol{x_2} \boxplus 3\boldsymbol{x_1} \boxplus \boldsymbol{x_0})(3\boldsymbol{a_3} \boxplus 2\boldsymbol{a_2} \boxplus 3\boldsymbol{a_1} \boxplus \boldsymbol{a_0}) \\ (3\boldsymbol{x_3} \boxplus 3\boldsymbol{x_2} \boxplus \boldsymbol{x_1} \boxplus 2\boldsymbol{x_0})(3\boldsymbol{a_3} \boxplus 3\boldsymbol{a_2} \boxplus \boldsymbol{a_1} \boxplus 2\boldsymbol{a_0}) \\ (2\boldsymbol{x_3} \boxplus \boldsymbol{x_2} \boxplus \boldsymbol{x_1} \boxplus \boldsymbol{x_0})(2\boldsymbol{a_3} \boxplus \boldsymbol{a_2} \boxplus \boldsymbol{a_1} \boxplus \boldsymbol{a_0}) \\ (3\boldsymbol{x_3} \boxplus \boldsymbol{x_2} \boxplus 2\boldsymbol{x_1} \boxplus 3\boldsymbol{x_0})(3\boldsymbol{a_3} \boxplus \boldsymbol{a_2} \boxplus 2\boldsymbol{a_1} \boxplus 3\boldsymbol{a_0}) \end{bmatrix},
\end{aligned}
$$

which is a linear transformation of $x$ over $\mathbb{Z}_4$ with matrix

$$
\begin{bmatrix}
2\boldsymbol{a_1} \boxplus 2\boldsymbol{a_0} & 0 & 2\boldsymbol{a_3} \boxplus 2\boldsymbol{a_0} & 2\boldsymbol{a_3} \boxplus 2\boldsymbol{a_1} \\
2\boldsymbol{a_2} \boxplus 2\boldsymbol{a_1} & 2\boldsymbol{a_3} \boxplus 2\boldsymbol{a_1} & 2\boldsymbol{a_3} \boxplus 2\boldsymbol{a_2} & 0 \\
0 & 2\boldsymbol{a_1} \boxplus 2\boldsymbol{a_0} & 2\boldsymbol{a_2} \boxplus 2\boldsymbol{a_0} & 2\boldsymbol{a_2} \boxplus 2\boldsymbol{a_1} \\
2\boldsymbol{a_2} \boxplus 2\boldsymbol{a_0} & 2\boldsymbol{a_3} \boxplus 2\boldsymbol{a_0} & 0 & 2\boldsymbol{a_3} \boxplus 2\boldsymbol{a_2}
\end{bmatrix}.
$$

Since the sum of all the rows is zero, the rank of the matrix is at most 3. Moreover, since there is at least one $A_i$ which equals 0 or 2, due to the assumption that $\hat{\boldsymbol{a}} \in \Lambda_i, 1 \leq i \leq 14$, it can be easily verified that the rank of the matrix is exactly 2. Therefore, there are four possible values that $\Phi(M\hat{\boldsymbol{f}}) \oplus \hat{e}$ can take:

$$
\begin{aligned}
& \Phi(\begin{bmatrix} 2\boldsymbol{a_1} \boxplus 2\boldsymbol{a_0}, & 2\boldsymbol{a_2} \boxplus 2\boldsymbol{a_1}, & 0, & 2\boldsymbol{a_2} \boxplus 2\boldsymbol{a_0} \end{bmatrix}^T) \\
& \Phi(\begin{bmatrix} 0, & 2\boldsymbol{a_3} \boxplus 2\boldsymbol{a_1}, & 2\boldsymbol{a_1} \boxplus 2\boldsymbol{a_0}, & 2\boldsymbol{a_3} \boxplus 2\boldsymbol{a_0} \end{bmatrix}^T) \\
& \Phi(\begin{bmatrix} 2\boldsymbol{a_3} \boxplus 2\boldsymbol{a_0}, & 2\boldsymbol{a_3} \boxplus 2\boldsymbol{a_2}, & 2\boldsymbol{a_2} \boxplus 2\boldsymbol{a_0}, & 0 \end{bmatrix}^T) \\
& \Phi(\begin{bmatrix} 2\boldsymbol{a_3} \boxplus 2\boldsymbol{a_1}, & 0, & 2\boldsymbol{a_2} \boxplus 2\boldsymbol{a_1}, & 2\boldsymbol{a_3} \boxplus 2\boldsymbol{a_2} \end{bmatrix}^T)
\end{aligned},
$$

if $\hat{\boldsymbol{a}} \in \Lambda_i, i = 1, 2, 4, 7, 8, 11, 13$, or the linear span of the above four vectors if $\hat{\boldsymbol{a}} \in \Lambda_i, i = 3, 5, 6, 9, 10, 12$ (due to that these four expressions only evaluate to two different vectors). Each value appears with probability 1/4. That is to say, for any input difference $\hat{a}$ with $\Phi^{-1}(\hat{a}) \in \Lambda_i, 1 \leq i \leq 14$, there are four output differences, and the differential probability is 1/4, which concludes the proof. □

**Remark 4.** *Let $D_0$ be an input difference with four output differences $D_1, D_2, D_3, D_4$. Then, the output differences propagate to the same set of four differences through $\zeta$. For instance, the input difference* `01` *has the output differences* `76, 79, B5, BA`*, and they have the same output differences* `40, 4F, 73, 7C` *when acting as input difference to the function $\zeta$. We call the propagation of differences* {`76, 79, B5, BA`} *to* {`40, 4F, 73, 7C`} *a package. Note*

*that the propagations with probability* 1 *are packages naturally. Furthermore, it can be easily verified with the expressions of the output differences that the packages can be categorised into loops. For instance, the following four packages form a loop of length* 4*:*

$$\{\texttt{01}, \texttt{3D}, \texttt{CD}, \texttt{F1}\} \rightarrow \{\texttt{76}, \texttt{79}, \texttt{B5}, \texttt{BA}\}\,,$$

$$\{\texttt{76}, \texttt{79}, \texttt{B5}, \texttt{BA}\} \rightarrow \{\texttt{40}, \texttt{4F}, \texttt{73}, \texttt{7C}\}\,,$$

$$\{\texttt{40}, \texttt{4F}, \texttt{73}, \texttt{7C}\} \rightarrow \{\texttt{5E}, \texttt{6D}, \texttt{9D}, \texttt{AE}\}\,,$$

$$\{\texttt{5E}, \texttt{6D}, \texttt{9D}, \texttt{AE}\} \rightarrow \{\texttt{01}, \texttt{3D}, \texttt{CD}, \texttt{F1}\}\,.$$

## 5.4 A General Construction of Nonlinear Diffusion Functions Based on T-functions

In order to find a systematic nonlinear code $(2n, n, d)$ over a finite field $\mathbb{F}_{2^m}$, one may start with identifying an appropriate nonlinear bijection $f : \mathbb{F}_{2^m}^n \rightarrow \mathbb{F}_{2^m}^n$, such that the codewords are of the form:

$$\hat{c} = (\hat{x}, f(\hat{x})),\ \hat{x} \in \mathbb{F}_{2^m}^n\,.$$

However, there are no guidelines for the choice of the nonlinear functions due to the lack of the generator matrix. Given a certain nonlinear function, it is also time-consuming to find the minimal distance and the weight distribution of the code, since this requires to generate all the codewords and computing the differences of all pairs.

In this section, we consider a general family of nonlinear functions based on the T-functions. The original idea of this construction came from the observations on ARX structures and automatons. We would like to thank Gregor Leander who came up with an elegant proof of the cryptographic properties and further generalised the constructions during fruitful discussions.

Let $k, l \geq 0$ be integers. Let $\eta : \mathbb{F}_{2^k}^l \rightarrow \mathbb{F}_{2^k}^l$ be a nonlinear function

$$\hat{y} = \eta(\hat{x})\,,$$

where $\hat{y} = (y_{l-1}, \ldots, y_1, y_0)^T$ and $\hat{x} = (x_{l-1}, \ldots, x_1, x_0)^T$. We define $\eta$ to be a function such that the $j$-th bit of each of the $y_i$

1. depends linearly on the $j$-th bits of $x_\ell$, *i.e.*, $x_\ell[j]$ for all $1 \leq \ell \leq l$.

2. depends in an arbitrary way on the $t$-th bits of $x_\ell$ for any $1 \le \ell \le l$ and for $t < j$
3. does not depend on the $t$-th bits of $x_\ell$ for all $1 \le \ell \le l$ and for $t > j$

Denote $\hat{x}^{(j)} = (x_{l-1}[j], \ldots, x_0[j])^T$ and $\hat{y}^{(j)} = (y_{l-1}[j], \ldots, y_0[j])^T$. For all $0 \le j \le k-1$, there exist a function

$$T_j : \left(\mathbb{F}_2^l\right)^j \to \mathbb{F}_2^l$$

and a linear function $m_j$

$$m_j : \mathbb{F}_2^l \to \mathbb{F}_2^l$$

defined by a matrix $M_j \in \mathbb{F}_2^{l \times l}$ such that

$$\hat{y}^{(j)} = M_j \hat{x}^{(j)} + T_j\left(\hat{x}^{(j-1)}, \ldots, \hat{x}^{(0)}\right).$$

**Theorem 9.** *Let $\eta$ be a function defined as above. Then $\eta$ is a permutation if all the matrices $M_j$ are invertible. Furthermore, the branch number of $\eta$ is*

$$\min_{0 \le j \le k-1} B_d(M_j) \le B_d(\eta) \le \max_{0 \le j \le k-1} B_d(M_j).$$

*Proof.* Suppose that

$$\hat{y}^{(j)} = M_j \hat{x}^{(j)} \oplus T_j(\hat{x}^{(j-1)}, \ldots, \hat{x}^{(0)}),$$

and

$$\hat{y}^{(j)} \oplus \Delta\hat{y}^{(j)} = M_j(\hat{x}^{(j)} \oplus \Delta\hat{x}^{(j)}) \oplus T_j(\hat{x}^{(j-1)} \oplus \Delta\hat{x}^{(j-1)}, \ldots, \hat{x}^{(0)} \oplus \Delta\hat{x}^{(0)}.$$

We consider $j_0 \ge 0$ where $\Delta\hat{x}^{(j_0)}$ is the first nonzero component of $(\Delta\hat{x}^{(t-1)}, \ldots, \Delta\hat{x}^{(0)})^T$. Then, one has $\Delta\hat{x}^{(j-1)} = 0$ for $j < j_0$. Therefore,

$$T_j(\hat{x}^{(j_0-1)}, \ldots, \hat{x}^{(0)}) = T_j(\hat{x}^{(j_0-1)} \oplus \Delta\hat{x}^{(j_0-1)}, \ldots, \hat{x}^{(0)} \oplus \Delta\hat{x}^{(0)}),$$

which means

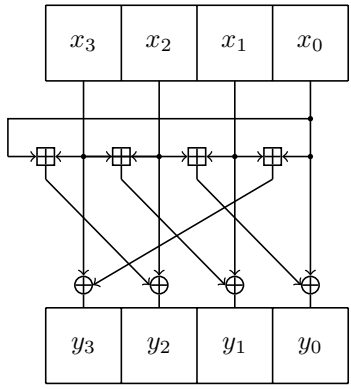
$$\Delta\hat{y}^{(j_0)} = M_{j_0} \Delta\hat{x}^{(j_0)}.$$

**Invertible:** When $(\Delta\hat{y}^{(k-1)}, \Delta\hat{y}^{(k-2)}, \ldots, \Delta\hat{y}^{(0)})^T = 0$, in particular, $\Delta\hat{y}^{(j_0)} = 0$. Since every $M_j$ is invertible, we have $\Delta\hat{x}^{(j_0)} = 0$. By induction, we get $(\Delta\hat{x}^{(k-1)}, \Delta\hat{x}^{(k-2)}, \ldots, \Delta\hat{x}^{(0)})^T = 0$.

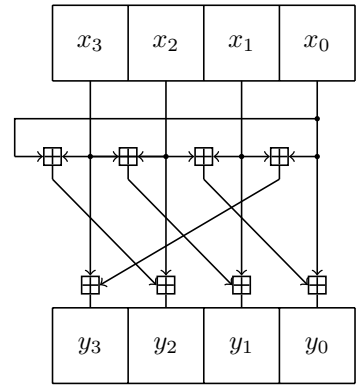
**Branch number:** When there are $\alpha$ nonzero bits in $\Delta\hat{y}^{(j_0)}$, *i.e.*, the Hamming weight of $\Delta\hat{y}^{(j_0)}$ is $wt(\Delta\hat{y}^{(j_0)}) = \alpha$, we know that $wt(\Delta\hat{y}^{(j_0)}) + wt(\Delta\hat{x}^{(j_0)}) \ge B_d(M_{j_0})$. Therefore, the branch number of function $\eta$:

$$\min_{0 \le j \le k-1} B_d(M_j) \le B_d(\eta) \le \max_{0 \le j \le k-1} B_d(M_j).$$

□




(a)a nonlinear function $\rho$




(b)a nonlinear function $\psi$

Figure 5.1: (a): a nonlinear function with XOR and modular additions, and (b): a nonlinear function with only modular additions.

With the generic construction, one can adjust the nonlinear diffusion function according to the goal of the design, meanwhile the diffusion property of the function is fully under control. As an application, we define two nonlinear diffusion functions based on modular addition.

### 5.4.1 The Nonlinear Diffusion Functions $\rho$ and $\psi$

Let $m \geq 2$ be an integer. In most cases, $m$ equals 4 or 8 according to the size of the S-box in use. We define a nonlinear function over $\mathbb{F}_{2^m}$ as

$$\rho : \mathbb{F}_{2^m}^4 \to \mathbb{F}_{2^m}^4 : (x_3, x_2, x_1, x_0) \mapsto (y_3, y_2, y_1, y_0)\,,$$

where $y_i = x_i \oplus (x_{i+1} \boxplus x_{i+2})$, $0 \leq i \leq 3$. Figure 5.1(a) gives a picture of $\rho$.

By slightly modifying the function $\rho$, we define a nonlinear function $\psi$ which is shown in Figure 5.1(b):

$$\psi : \mathbb{F}_{2^m}^4 \to \mathbb{F}_{2^m}^4 : (x_3, x_2, x_1, x_0) \mapsto (y_3, y_2, y_1, y_0)\,,$$

where $y_i = x_i \boxplus (x_{i+1} \boxplus x_{i+2})$, $0 \leq i \leq 3$.

### 5.4.2 Diffusion Functions $\rho$ and $\psi$

As special instances of the general construction, the branch number of the nonlinear functions $\rho$ and $\psi$ can be predicted accordingly.

**Theorem 10.** *The nonlinear function $\rho$ is invertible and has differential branch number 4.*

*Proof.* The modular addition of two elements $x \boxplus y$ can be written as $x \oplus y \oplus c(x, y)$, where the carry function $c(\cdot, \cdot)$ is a T-function. As a consequence, the update function of $\rho$,

$$y_i = x_i \oplus (x_{i+1} \boxplus x_{i+2})$$

is equivalent to

$$y_i = x_i \oplus x_{i+1} \oplus x_{i+2} \oplus c(x_{i+1}, x_{i+2}),$$

which follows the general construction. The matrix representation for the linear part of the update function is

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}, \tag{5.9}$$

which has branch number 4. Hence, the nonlinear function is invertible with differential branch number 4. □

**Remark 5.** *Note that $\rho$ is similar to the S-box ($\chi$ function) adopted in the SHA-3 permutation* Keccak-$f$ *[30], which is a generalised automaton [61, 206]. Although $\rho$ is invertible, the inverse of $\rho$ seems to not have a simple expression.*

The property of the nonlinear function $\psi$ can be analysed analogously to that of $\rho$. Here, notice that the operations in the nonlinear function $\psi$ are only modular additions, hence, we take a slightly different approach to show the property of $\psi$ as follows.

**Theorem 11.** *$\psi$ is invertible and has differential branch number* 4.

*Proof.* Although $\psi$ is a nonlinear function over $\mathbb{F}_{2^m}$, it is linear over the ring of integer modulo $2^m$, *i.e.*, $\mathbb{Z}_{2^m}$. Therefore, it has a generator matrix over $\mathbb{Z}_{2^m}$:

$$\mathbf{\Psi} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix},$$

which is near-MDS. As a result, $\psi$ is invertible and its differential branch number is 4. □

Due to the linearity over $\mathbb{Z}_{2^m}$, the inverse of $\psi$ has a simple form with generator matrix $\mathbf{\Psi}^{-1}$. When $m = 4$, the update rule of $\psi^{-1}$ is defined by the generator matrix:

$$\mathbf{\Psi}^{-1} = \begin{pmatrix} 10 & 11 & 11 & 11 \\ 11 & 11 & 11 & 10 \\ 11 & 11 & 10 & 11 \\ 11 & 10 & 11 & 11 \end{pmatrix}.$$

**Remark 6.** *It is interesting to notice that the linear part of the functions $\rho$ and $\psi$ is affine-equivalent to a well-known linear function $\sigma : \mathbb{F}_{2^m}^4 \to \mathbb{F}_{2^m}^4$ defined by the binary near-MDS matrix*

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}. \tag{5.10}$$

*The function $\sigma$ has been successfully applied to lightweight block cipher designs such as* Midori *[12], and the similarity suggests possible applications of the nonlinear functions $\rho$ and $\psi$.*

## 5.5 Example Ciphers with a Nonlinear Diffusion Layer

### 5.5.1 Nonlinear Diffusion Function $\zeta$

The diffusion layer based on the Kerdock nonlinear function operates on bits. In order to fully utilise the diffusion property, we provide here an example cipher with 128-bit block size as shown in Figure 5.2. The 128-bit block is arranged as a 3-dimensional state with rows, columns and lanes of length 4, 8, and 4, respectively. The cipher has two different round functions. For odd-numbered rounds, the round function is defined as follows:

- SubCells: 32 applications of a 4-bit S-box (the S-box of Present [44]) on each row

$$S[16] = \{\mathtt{C}, 5, 6, \mathtt{B}, 9, 0, \mathtt{A}, \mathtt{D}, 3, \mathtt{E}, \mathtt{F}, 8, 4, 7, 1, 2\};$$

- MixColumns: 16 applications of the nonlinear map $\zeta$ derived from $\mathcal{K}(4)$ on each column;
- AddKey: a 128-bit key is XORed into the state,

and for even-numbered rounds it is as follows:

- SubCells: 32 applications of the S-box on each row;
- MixLanes: 32 applications on each lanes of the linear function $\sigma : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ in Equation (5.10);
- AddKey: a 128-bit key is XORed into the state.

To evaluate the bound for the number of active S-boxes in the cipher, we performed an automatic search of differential characteristics, adopting a similar method as [189]. Denote the 4-bit differences before and after each SubCells layer by $a_i^r$ and $b_i^r$, where $0 \leq r \leq n$ stands for the round number and $0 \leq i \leq 15$ labels the S-boxes. Model the propagation of $(a_i^r \to b_i^r)$ by the DDT table of the S-box. For odd-numbered rounds, the difference propagation of $(b_i^r \to a_i^{r+1})$ follows Theorem 8. For even-numbered rounds, the MixLanes operation is linear, therefore, the propagation of differences is straightforward.

It is shown by automatic search that the minimum number of active S-boxes in a 4-round differential characteristic is 24, while the number is 20 with a linear diffusion layer where the branch number is at most 5. Furthermore, the probability of a differential characteristic with optimal number of active S-boxes can be much lower than what is provided by the S-boxes solely. For instance, we show a 4-round characteristic with 24 active S-boxes in Figure 5.3. To simplify the view, the differences over the cells are 4-bit long which represent the hexadecimal values of the bits in a row of the cipher, and a blank cell stands for a zero difference.

The 24 active S-boxes lead to a differential probability of $2^{-61}$ (taking into account the propagations of probability $2^{-3}$). Moreover, we define the nonlinear diffusion functions being *effectively active* if the difference propagations have nonzero probability which is strictly less than 1 (the propagations with probability 1 have no effective contribution). In our example, there are 8 effectively active nonlinear diffusion functions, which counts for an additional probability of $2^{-16}$. Therefore, the overall expected differential probability of the characteristic is $2^{-77}$.

**Remark 7.** *The cipher we propose here has some similarity to previous designs such as the LS-designs [96] which are* 2*-dimension and the block cipher 3D [162].*

### 5.5.2 Nonlinear Diffusion Function $\rho$ and $\psi$

The nonlinear functions $\rho$ and $\psi$ operate on $\mathbb{F}_{2^m}^4$, therefore, they can be adopted similarly as most MixColumns operations in the AES-like designs. We illustrate

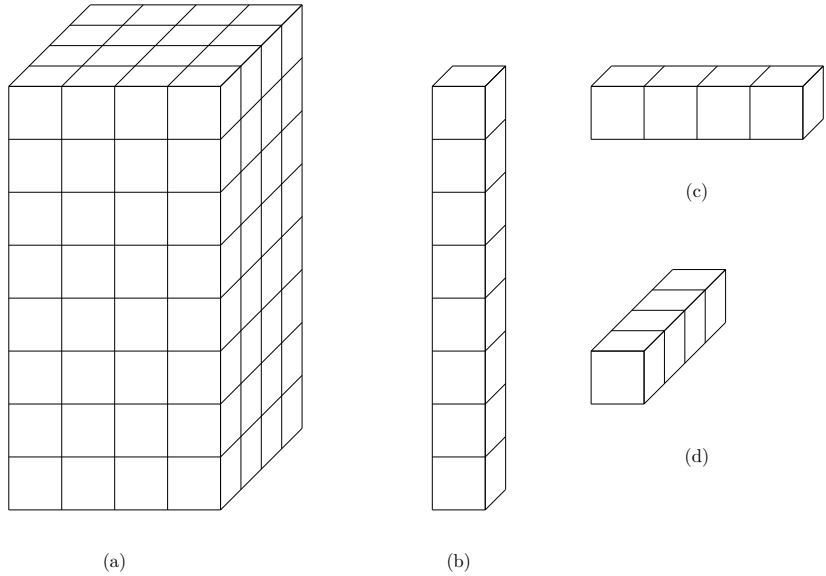



Figure 5.2: (a) An example cipher of 128 bits, (b) a column, (c) a row, (d) a lane.

an application of the nonlinear diffusion function $\rho$ in the following 64-bit cipher. The state is arranged in a 4×4 square

$$\begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix}.$$

The round function consists the following operations:

- SubCells: a 4-bit S-box (the S-box of PRINCE) is applied to each cell;
- ShiftBits: the value on each column is rotated to the left by $r$ bits, *i.e.*,

  $$(s_{4i}, s_{4i+1}, s_{4i+2}, s_{4i+3}) \leftarrow ((s_{4i}, s_{4i+1}, s_{4i+2}, s_{4i+3}) \lll r), i = 0, 1, 2, 3\,;$$

- ShuffleCells: the same operation as that of MIDORI, *i.e.*,

  $$(s_0, s_1, ..., s_{15}) \leftarrow (s_0, s_{10}, s_5, s_{15}, s_{14}, s_4, s_{11}, s_1, s_9, s_3, s_{12}, s_6, s_7, s_{13}, s_2, s_8)\,;$$

- MixColumns: $\rho$ is applied to each column of the state;

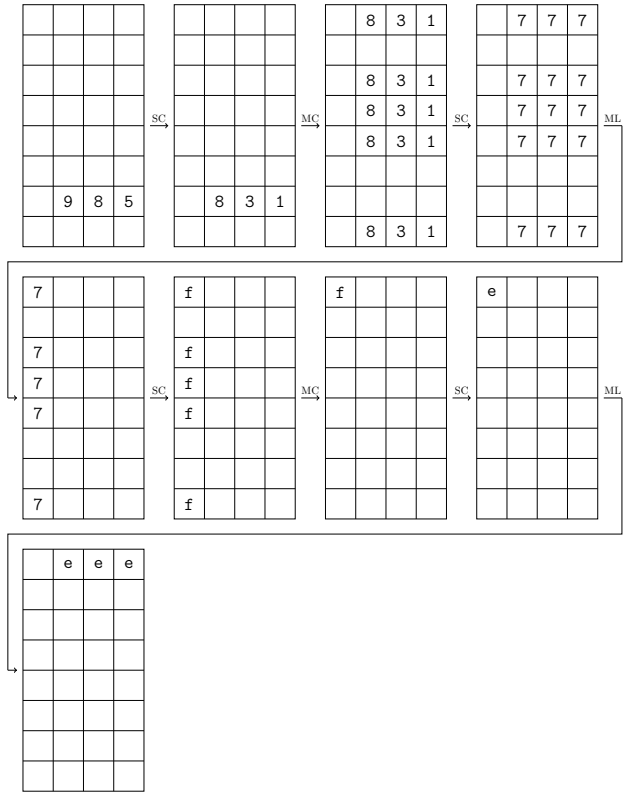



Figure 5.3: A 4-round differential characteristic with 24 active S-boxes.

- KeyAdd: a 64-bit key is XORed to the state.

Despite its similarity to MIDORI, the cipher we propose here has the operation ShiftBits, which is uncommon for S-box-based ciphers. The main target of ShiftBits is to avoid trivial propagations of differences through the MixColumns operation where the modular additions are not effectively activated. The difference propagation through modular addition can be efficiently modelled with a SAT/SMT language similar to the techniques in [150, 130]. As a result, we track the propagation of the differences through the round function, and obtain that when the rotational offset $r$ is 3, an optimal differential characteristic of 4 rounds has probability $2^{-42}$. Compared to MIDORI64 with at least 16 active S-boxes after 4 rounds leading to a differential probability $2^{-32}$ [12], the modular additions of $\rho$ contribute significantly to the resistance against differential cryptanalysis.

**Remark 8.** *The main focus of these example ciphers is the nonlinear diffusion layer, therefore, we ignore the specification of their key schedule. By experiment, one can show that the linear branch number of the function* $\zeta$ *is* 6 *and those of* $\rho, \psi$ *are* 4 *when operating over* 4*-bit words. Hence, for simplicity and illustrative purposes, we only concentrate on analysing differential characteristics of the ciphers.*

## 5.6 Conclusions

In this chapter, we proposed several nonlinear functions with good diffusion, which can be adopted as a replacement of the linear layers in block ciphers; our nonlinear functions provide both diffusion and confusion. We investigate a nonlinear function derived from the Kerdock codes, and two nonlinear functions of an automaton structure based on modular additions, which are derived from a generic construction of nonlinear functions based on T-functions proposed in this chapter. We proved that the diffusion property in terms of the branch number has a comparable or even better effect compared to their linear counterparts. To show an application of nonlinear diffusion layers, the diffusion effect of the nonlinear function is illustrated in two example ciphers. Due to the improved diffusion of the $\zeta$ function, a higher bound on the minimum number of the active S-boxes is guaranteed. Moreover, the confusion effect is further boosted by the probabilistic propagations of differences through the nonlinear diffusion layer.

A future line of work could focus on constructing nonlinear diffusion functions suitable for various designing preferences, as well as a better trade-off between the diffusion and confusion effect of a nonlinear diffusion function when interacting with the S-boxes and (possibly also) the linear functions. In terms of the resistance against differential and linear cryptanalysis, it is promising to design lightweight ciphers which reach the security requirement with fewer rounds.

In addition, designing ciphers for MPC and FHE benefits from the nonlinear diffusion layers, since fewer rounds may reduce the overall multiplicative depth. For side-channel resistance, masking schemes such as the Threshold Implementation [165] prefer operations with a lower algebraic degree; in this case, nonlinear diffusion functions based on bitwise AND can be advantageous.

# Chapter 6

# The Phantom of Differential Characteristics

As we have seen in previous chapters, statistical distinguishers are extensively studied and applied to symmetric cryptanalysis. Those distinguishers are based on several fundamental assumptions. In this chapter, our focus is on the accuracy of the assumptions when considering practical ciphers, especially the influence on the characteristics in differential cryptanalysis.

This chapter is organised as follows. Section 6.1 motivates the problem we will study and reviews previous works. We show our basic observation on effective keys of the characteristics in Section 6.2. The singular characteristic as well as singular cluster are proposed in Section 6.3. Section 6.4 shows the properties of singular characteristics and singular clusters in the AES. Section 6.5 further discusses the different views of the designer and the attacker with respect to the security of a block cipher. Finally, we conclude in Section 6.6.

## 6.1 Motivation

Symmetric-key designs are expected to resist known cryptanalytic methods based on distinguishers that distinguish a (round-reduced) block cipher from random permutations. Introduced by Biham and Shamir in 1990, differential cryptanalysis has been successfully applied to the analysis of numerous block ciphers and hash functions, see for instance [36, 115]. In a single-key setting, the difference in a block cipher comes from the plaintexts, which play the same

role as the chaining values in a hash function; while the difference in a hash function is injected through the messages, which resembles the round keys in a block cipher. Generally speaking, finding collisions for a hash function is somehow analogous to searching for related-key differentials of a block cipher, examples are the collision attacks on MD5 and SHA-1 [184, 202, 203], as well as the rebound attacks on the SHA-3 candidates [156].

**Key Schedule.** A block cipher is designed with a key schedule which expands the key into several round keys. We often assume that independent keys are produced in successive rounds, meanwhile the key schedule cannot be too complicated to reduce the implementation cost. Therefore, in practical block cipher designs, the operations of a key schedule are mostly linear. For example in the key schedule of AES-128, only 4 out of 16 bytes pass through the S-box operations. Meanwhile, linear key schedules are commonly adopted, especially in many lightweight ciphers.

The key schedules are hardly explored to construct a distinguisher for a (round-reduced) cipher. Indeed, cryptanalysis typically searches for the distinguishing properties that are independent of the key schedules. For example, truncated impossible differentials independent of the non-linear components and the key schedule [34, 187] are constructed in most impossible differential attacks. Even if they have been used to mount attacks in the related-key setting [37], the key schedules are in most cases merely involved in the key recovery attack to reduce the guessing complexity by exploring the relations among the round keys. As a matter of fact, key schedules are considered irrelevant to the distinguishers themselves in the single-key setting, such as the search of differential characteristics with an automated tool [188].

The overlooking of the key schedules leads to a bottle-neck in finding better distinguishers. As shown in a previous research [186], unless the details of the S-boxes and the key schedule are explored, the 4-round impossible differentials of the AES cannot be improved. Unfortunately though, the role of the key schedule in extending the distinguishers under the single-key model still remains an open problem.

**Hypothesis of Stochastic Equivalence.** The probability of a differential is the sum of the probabilities of the corresponding differential characteristics. Hence, an estimation on the probabilities of differential characteristics is crucial to the validity of an analysis. However, predicting the probability for a given differential characteristic under a specific key schedule is known to be a difficult problem. A widely accepted solution is to assume that the probability varies negligibly for different keys, which is the hypothesis of stochastic equivalence [132], see also section 2.2.1. Meanwhile, the primitive is assumed to be a Markov cipher and the round keys are independent and

uniformly distributed despite the key schedule. Under these assumptions, the probability of a differential characteristic is estimated by the product of the probability in each round, which is the expected differential probability (EDP) of the characteristic, *i.e.*, the averaged probability over all independent round keys. The assumptions enable the designers to provide security proofs against differential cryptanalysis by bounding the minimum number of active S-boxes, such as the wide trail strategy in the design of the AES [64], as we have also implicitly utilised in previous chapters. Currently, this model serves as the main methodology of evaluating block ciphers against differential cryptanalysis.

Even though the hypothesis of stochastic equivalence and the Markov model provide reliable bounds for the designers, they encounter exceptions from the point of view of attackers [103, 123, 200]. For instance, there exists a discrepancy between the experiments and the theoretical estimation in a chain of modular additions for differential cryptanalysis and rotational cryptanalysis [118, 139]. In the meantime, the existence of 2-round plateau characteristics in the AES indicates the mismatch of fixed-key and the expected differential probability in some SPN block ciphers [66]. Therefore, it is vital to test whether a differential characteristic with a non-zero EDP is a real differential characteristic in a block cipher with a specific key schedule. As a direct consequence, some differential attacks on block ciphers may be at stake. Moreover, if the characteristics can be shown to be of probability zero, an attack will probably fail since the techniques for the key recovery attack based on a differential and an impossible differential are essentially different.

Consider the following toy cipher. Let $S(\cdot)$ be the 8-bit S-box of the AES and $S_k^r(x) = S(S_k^{r-1}(x) \oplus k)$, where $S_k^1(x) = S(x \oplus k)$. For a fixed-key $k$, one can construct at most $\binom{256}{2} \approx 2^{15}$ differential characteristics of $S_k^r(x)$ by naming the pairs of plaintexts. However, under the hypothesis of stochastic equivalence and the assumption of the Markov model, since for each input difference of the S-box, there are about $2^7$ possible output differences, we can find $2^8 \times 2^7 \times 2^7 \cdots \times 2^7 = 2^{7r+8}$ differential characteristics with nonzero probability. Thus, for any characteristic, its probability of being valid is approximately $2^{8+15}/2^{7r+8} = 2^{-(7r-15)}$, which is marginal when $r$ is large. From this point of view, it is probable that a differential characteristic with a nonzero EDP may turn out to be an impossible one. Therefore, the results of characteristic-based differential cryptanalysis may be suspicious unless one can claim the characteristic is a real one.

## 6.2 Basic Observation

Throughout this chapter, we are only interested in whether a differential characteristic is a real one or not, and we will neglect the value of a nonzero probability. For a fixed permutation, the *character* of a differential characteristic $\Omega$ is defined as follows:

$$\chi(\Omega) = \begin{cases} 1 & \Pr(\Omega) > 0, \\ 0 & \Pr(\Omega) = 0. \end{cases}$$

The character function distinguishes whether a differential characteristic is *possible* or not. Furthermore, let $E$ be a block cipher and $k$ be a fixed key. If we can find a pair of plaintexts such that the characteristic $\Omega$ holds, we denote $\chi_k(\Omega) = 1$. Otherwise, $\chi_k(\Omega) = 0$. For a vectorial Boolean function $f$, if $f(x \oplus \delta) \oplus f(x) = \Delta$, we call $x$ a *right input* of $\delta \to \Delta$ and we denote by $RI(\delta, \Delta)$ all the right inputs of $\delta \to \Delta$. Similarly, we denote by $RO(\delta, \Delta)$ all the right outputs of $\delta \to \Delta$. Obviously, one has

$$RO(\delta, \Delta) = \{y = f(x) \,|\, x \in RI(\delta, \Delta)\}\,.$$

As illustrated before, a randomly-constructed characteristic under the hypothesis of stochastic equivalence may well be invalid in reality. If a differential characteristic with nonzero EDP is not a real characteristic in a block cipher, any key recovery attack based on such a distinguisher will fail. A probably more urgent concern is that the key schedule has a fundamental influence on the validity of the characteristics, contrary to the fact that key schedules have been ignored for a long time. For example, the AES encompasses three different key schedules to support variable key size. However in practice, a differential characteristic of AES-128 is naturally considered as a characteristic of AES-192 due to the same underlying round function.

Consider an $r$-round characteristic $\Omega : \alpha_0 \xrightarrow{S} \beta_0 \xrightarrow{P} \cdots \xrightarrow{P} \alpha_{r-1} \xrightarrow{S} \beta_{r-1}$ of an SPN cipher where $\alpha_{i+1} = P\beta_i, i = 0, \ldots, r-2$ and the round keys are XORed after the permutation layer. The XOR of the round keys does not affect the differences, but it changes the intermediate values for sure. Consequently, the actual number of right pairs following such a characteristic may largely vary with the round keys. Denote the output $y$ of the $i$-th S-box-layer by $RO_S(\alpha_{i-1}, \beta_{i-1})$, the input $z$ to the $(i+1)$-st S-box-layer by $RI_S(\alpha_i, \beta_i)$, and the $i$-th round key by $K_i$, respectively. The characteristic holds with a nonzero probability if $z = Py \oplus K_i$, which implies that

$$K_i \in [P \cdot RO_S(\alpha_{i-1}, \beta_{i-1})] \oplus RI_S(\alpha_i, \beta_i)\,. \tag{6.1}$$

The observation of Equation (6.1) is based on SPN ciphers; nevertheless, it can be extended to other structures as well. Thus we have the following claim:

**Claim.** A differential characteristic of a block cipher always corresponds to some keys, *i.e.*, the whole key space can be divided into two subsets $K_0$ and $K_1 = \overline{K}_0$ such that $\chi_k(\Omega) = 1$ if and only if $k \in K_0$.

It has already been noticed by previous studies that a differential characteristic may be of zero probability under certain keys, for instance, the plateau characteristics in the AES. Recall that the character function $\chi_k(\Omega)$ indicates whether for a given key $k$ the probability of a characteristic $\Omega$ is zero. We introduce the following definition of the *effective keys*.

**Definition 22.** *The keys for which a differential characteristic $\Omega$ holds with nonzero probability are called the effective keys of $\Omega$. The set containing all effective keys of the characteristic $\Omega$ is denoted by $K_\Omega$, i.e., $\chi_k(\Omega) = 1$ if and only if $k \in K_\Omega$. For differential characteristics $\Omega_0, \ldots, \Omega_{t-1}$, their effective keys are defined as the intersection of all $K_{\Omega_i}$, i.e., $K_{\Omega_0,\ldots,\Omega_{t-1}} = \cap_{i=0}^{t-1} K_{\Omega_i}$.*

Usually, due to the complex relation between plaintexts and the keys, as well as a detailed key schedule, computing the effective keys of a differential characteristic is difficult. However, if the S-box is differentially 4-uniform, *i.e.*, the entries in the DDT table of the S-box are 0, 2 or 4, we have the following theorem on the effective keys of a 2-round characteristic in an SPN cipher, which is also the behaviour of the keys observed in plateau characteristics [66].

**Theorem 12.** *Let $\mathcal{E}$ be a 2-round SPN cipher with differentially 4-uniform S-boxes and $\Omega$ be a differential characteristic of $\mathcal{E}$. Then, the effective keys $K_\Omega$ form a linear (affine) subspace.*

*Proof.* The key point of the proof is based on the fact that if the number of right pairs for the active S-boxes involved in a differential characteristic is either 1 or 2, the right inputs/outputs of each active S-box form a linear/affine subspace.

If there are two right inputs, say $\{a, a \oplus \delta\}$, we can further denote the right inputs by $a \oplus \{0, \delta\} = \{\delta x\} \oplus a$ where $x \in \mathbb{F}_2$. Thus the dimension of the linear subspace is 1. If there are four right inputs, say $\{a, a \oplus \delta, b, b \oplus \delta\}$, we can further denote the right inputs by $a \oplus \{0, \delta, a \oplus b, a \oplus b \oplus \delta\} = \{\delta x_1 \oplus (a \oplus b)x_2\} \oplus a$ where $x_1, x_2 \in \mathbb{F}_2$. Thus the dimension of the linear subspace is 2.

It is trivial that if an S-box is not active, the right inputs form exactly the whole space.

Then, according to Equation (6.1), the effective keys form the following set:

$$[P \cdot RO_S(\alpha_0, \beta_0)] \oplus RI_S(\alpha_1, \beta_1)\,.$$

Since $RO_S(\alpha_0, \beta_0)$ and $RI_S(\alpha_1, \beta_1)$ are linear/affine subspaces, the effective keys form a linear/affine subspace. □

As shown by our experiments, a diffusion layer with a simple algebraic structure, such as the MixColumns operation in the AES, leads to a rather small or even empty set of effective keys. In contrast, a 2-round characteristic of PRESENT is likely to possess a large amount of effective keys, as we observed in characteristics with low Hamming weights.

**Example 4.** *Let a 2-round characteristic of the AES be*

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{P} \begin{pmatrix} 2 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 3 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{pmatrix}.$$

*The right pairs of the differences propagating through the active S-boxes are* $RI_S(1,1) = \{\mathtt{CE}, \mathtt{CF}\} = \{\mathtt{CE} \oplus (x_0 \cdot 1) | x_0 \in \mathbb{F}_2\}$, $RO_S(1,1) = \{\mathtt{8A}, \mathtt{8B}\} = \{\mathtt{8A} \oplus (x_1 \cdot 1) | x_1 \in \mathbb{F}_2\}$, $RI_S(2,3) = \{10, 12\} = \{10 \oplus (x_2 \cdot 2) | x_2 \in \mathtt{F}_2\}$, $RI_S(3,2) = \{\mathtt{18}, \mathtt{1B}\} = \{\mathtt{18} \oplus (x_3 \cdot 3) | x_3 \in \mathbb{F}_2\}$. *The values passing through an inactive S-box can take any value in* $\mathbb{F}_{2^8}$*, which can be represented as a free variable of 8 bits. Hence, one gets that the first column of the effective keys falls into a linear subspace as follows:*

$$\left\{ \begin{pmatrix} 2 & m_3 & m_1 & m_1 & 2 & 0 & 0 & 0 \\ 1 & m_2 & m_3 & m_1 & 0 & 1 & 0 & 0 \\ 1 & m_1 & m_2 & m_3 & 0 & 0 & 1 & 0 \\ 3 & m_1 & m_1 & m_2 & 0 & 0 & 0 & 3 \end{pmatrix} \cdot x \oplus \begin{pmatrix} m_2 & m_3 & m_1 & m_1 \\ m_1 & m_2 & m_3 & m_1 \\ m_1 & m_1 & m_2 & m_3 \\ m_3 & m_1 & m_1 & m_2 \end{pmatrix} \begin{pmatrix} \mathtt{8A} \\ 0 \\ 0 \\ 0 \end{pmatrix} \oplus \begin{pmatrix} 10 \\ \mathtt{CE} \\ \mathtt{CE} \\ \mathtt{18} \end{pmatrix}, x \in \mathbb{F}_2^{29} \right\}.$$

*The hexadecimal numbers in the above linear system represent column vectors of 8 bits. The* MixColumns *matrix of the AES is denoted as*

$$\begin{pmatrix} m_2 & m_3 & m_1 & m_1 \\ m_1 & m_2 & m_3 & m_1 \\ m_1 & m_1 & m_2 & m_3 \\ m_3 & m_1 & m_1 & m_2 \end{pmatrix}$$

*with each* $m_i \in \mathbb{F}_2^{8\times 8}$ *being the matrix representation of the multiplication with* $i$ *over* $\mathbb{F}_2^8$*. To be specific,* $m_1$ *is the identity matrix and*

$$m_2 = \begin{pmatrix} 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 1&0&0&0&1&0&0&0\\ 1&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 1&0&0&0&0&0&0&1\\ 1&0&0&0&0&0&0&0 \end{pmatrix}, \quad m_3 = m_1 \oplus m_2 = \begin{pmatrix} 1&1&0&0&0&0&0&0\\ 0&1&1&0&0&0&0&0\\ 0&0&1&1&0&0&0&0\\ 1&0&0&1&1&0&0&0\\ 1&0&0&0&1&1&0&0\\ 0&0&0&0&0&1&1&0\\ 1&0&0&0&0&0&1&1\\ 1&0&0&0&0&0&0&1 \end{pmatrix}.$$

*This is also defined as the primitive representation [187] of a diffusion function. Meanwhile, the effective keys of the remaining columns have no constraints hence they can be any element of* $\mathbb{F}_2^{32}$*.*

**Remark 9.** *Our main focus here is on ciphers with differentially* 4*-uniform S-boxes, which covers a large number of block ciphers resembling the AES. Nonetheless, the method of identifying the effective keys is applicable to other S-boxes with a slight modification. For example, given an S-boxes with* 6 *right inputs for a certain difference propagation, the inputs can be classified into (affine) subspaces of dimension* 1 *or* 2*, which is reduced to the case in Theorem 12.*

## 6.3 Singular Characteristic and Singular Cluster

### 6.3.1 Singular Characteristic

Though the round keys are often assumed to be independently random for the sake of simplicity, all of them depend on the master key by the key schedule. For a key schedule with an $n$-bit master key generating $r$ round keys of $n$ bits each, the proportion of genuine keys out of all the independently random round keys is $2^{(1-r)n}$. Typically, the size of a round key is 128 bits or 64 bits, which means the fraction of genuine keys produced by the key schedule is marginal within all independently random round keys.

Recall that the set of effective keys of a differential characteristic is often a subset of all the possible keys. In this section, our focus is a seemingly peculiar but general phenomenon of characteristics, namely, the sets containing the effective keys for 2-round fragments of a characteristic are unfortunately inconsistent to each other with respect to the key schedule. We will demonstrate the existence of such "phantom" characteristics which have nonzero EDP while their probability is zero for all master keys.

**Definition 23.** *Let $\Omega$ be an $r$-round characteristic of a cipher $E$. If $K_\Omega = \emptyset$, we call $\Omega$ a singular characteristic.*

The difference between singular characteristics and the so-called impossible characteristics is that impossible characteristics are of probability zero due to the mismatch of difference propagations in the S-boxes or the linear layers, while a singular characteristic appears to be a characteristic if the information of the key schedule is not taken into consideration. Especially, techniques to enumerate differential characteristics with an automatic search may generate a large number of characteristic which are in fact singular, as we will show in Section 6.4. Usually, detecting a singular characteristic is not trivial. However, for those ciphers which have well-structured diffusion layers, we are able to mathematically describe the set of the effective keys. Furthermore, taking the key schedule into consideration, we execute the following strategy.

**Exploring the Key Schedule.**

Suppose that the key schedule updates the $(i+1)$-th round keys with the $i$-th one by $k_{i+1} = F(k_i)$, where $F$ is the key-expansion function. Consider a 3-round differential characteristic as follows:

$$\Omega : \alpha_0 \xrightarrow{S} \beta_0 \xrightarrow{P} P\beta_0 = \alpha_1 \xrightarrow{S} \beta_1 \xrightarrow{P} P\beta_1 = \alpha_2 \xrightarrow{S} \beta_2 \xrightarrow{P} P\beta_2 = \alpha_3.$$

According to Theorem 12, the effective keys of

$$\alpha_0 \xrightarrow{S} \beta_0 \xrightarrow{P} \alpha_1 \xrightarrow{S} \beta_1$$

can be written as a linear subspace, denote by $\mathcal{K}_1 = \{A_1 x_1 \oplus B_1\}$, where $A_1$ (resp. $B_1$) is a matrix (resp. vector) determined by the characteristic and $x_1$ is a vector of free variables. Similarly, denote the effective keys of

$$\alpha_1 \xrightarrow{S} \beta_1 \xrightarrow{P} \alpha_2 \xrightarrow{S} \beta_2$$

by $\mathcal{K}_2 = \{A_2 x_2 \oplus B_2\}$. According to the key schedule, we have that the keys satisfying $k_2 = F(k_1)$, for $k_1 \in \mathcal{K}_1, k_2 \in \mathcal{K}_2$, are the candidates of the effective keys for the 3-round characteristic.

For a general function $F$, we need to find the intersection of the sets $\mathcal{K}_2$ and $F(\mathcal{K}_1)$, in order to identify the effective keys. Notice that the key schedules in block ciphers tend to be light and simple comparing with the round functions, the problem can be much simplified if we place certain conditions on the function $F$. Assume that the key schedule only involves a few nonlinear operations, *i.e.*, the round keys satisfy $k_{i+1} = L \circ N(k_i)$, where the $N$ is a nonlinear function

that only applies to a small fraction of the bits in $k_i$ and the $L$ function is linear. Then, we have $k_2 = L \circ N(k_1)$, or equivalently, $L^{-1}(k_2) = N(k_1)$. Since $N$ only applies to a small fraction of the bits of $k_1$, $L^{-1}(k_2) = N(k_1)$ involves only a small number of non-linear equations. By deleting these non-linear equations from this system, we get a linear system $\mathbb{L}(k_1, k_2) = 0$ which could be reduced to $\mathbb{L}'(x_1, x_2) = 0$. If $\mathbb{L}'(x_1, x_2) = 0$ has no solutions, we can claim that the set of the effective keys is empty which implies that $\Omega$ is singular.

Note that the strategy for computing the effective keys of a 3-round differential characteristic can be extended to any number of rounds, as shown in Algorithm 5. As only the linear relations of the key schedule are utilised in the linear equation systems, the effective keys of a characteristic form a subset of the solutions of the equation system. As a result, our strategy may overlook the singularity of some characteristics. If there are only a few effective keys found by the equation system, it is possible to directly filter out the genuine keys for such characteristics with the key schedule.

**Algorithm 5** The algorithm to detect if a characteristic is singular

**Input:** An $r$-round characteristic $\Omega : \alpha_0 \xrightarrow{S} \beta_0 \xrightarrow{P} P\beta_0 = \alpha_1 \xrightarrow{S} \beta_1 \xrightarrow{P} P\beta_1 \ldots \alpha_{r-1} \xrightarrow{S} \beta_{r-1} \xrightarrow{P} P\beta_{r-1} = \alpha_r$
**Output:** The singularity of $\Omega$

1: Find effective keys for every 2-consecutive-round of $\Omega$ as $\mathcal{K}_i = \{A_i x_i \oplus B_i\}$, $1 \leq i \leq r-1$.
2: Build an equation system based on the key schedule: $k_{i+1} = L \circ N(k_i)$, $1 \leq i \leq r-2$, $k_i \in \mathcal{K}_i$.
3: Delete the nonlinear equations and get a linear equation system $\mathbb{L}(k_i, k_{i+1}) = 0$, $1 \leq i \leq r-2$.
4: Reduce the linear equation system into $\mathbb{L}'(x_i, x_{i+1}) = 0$, $1 \leq i \leq r-2$.
5: **if** Rank of coefficient matrix $\neq$ Rank of augment matrix **then**
6: **return** The characteristic $\Omega$ is singular.
7: **else**
8: **return** The singularity of $\Omega$ is undetermined.

### 6.3.2 Singular Clusters

When a differential contains only singular characteristics, it clearly has no effective keys. In a more general setting, considering two or more characteristics simultaneously, it is possible that the intersection of their effective keys turns

**Algorithm 6** The algorithm to detect if a pair of characteristics form a singular cluster

**Input:** $r$-round characteristics $\Omega : \alpha_0 \xrightarrow{S} \beta_0 \xrightarrow{P} P\beta_0 = \alpha_1 \xrightarrow{S} \beta_1 \xrightarrow{P} P\beta_1 \dots \alpha_{r-1} \xrightarrow{S} \beta_{r-1} \xrightarrow{P} P\beta_{r-1} = \alpha_r$ and $\Omega^* : \alpha_0^* \xrightarrow{S} \beta_0^* \xrightarrow{P} P\beta_0^* = \alpha_1^* \xrightarrow{S} \beta_1^* \xrightarrow{P} P\beta_1^* \dots \alpha_{r-1}^* \xrightarrow{S} \beta_{r-1}^* \xrightarrow{P} P\beta_{r-1}^* = \alpha_r^*$
**Output:** The singularity of the collection $\{\Omega, \Omega^*\}$

1: Find effective keys for every 2-consecutive-round of $\Omega$ and $\Omega^*$ as $\mathcal{K}_i = \{A_i x_i \oplus B_i\}$, and $\mathcal{K}_i^* = \{A_i^* x_i^* \oplus B_i^*\}$, respectively, for $1 \le i \le r-1$
2: Build a linear equation system with $k_i = k_i^*$, $1 \le i \le r-2$, $k_i \in \mathcal{K}_i$, $k_i^* \in \mathcal{K}_i^*$.
3: **if** Rank of coefficient matrix $\neq$ Rank of augment matrix **then**
4: **return** The collection is a singular cluster.
5: **else**
6: **return** The collection is undetermined.

out to be an empty set. In such case, the collection of these characteristics will have no effective key.

**Definition 24.** *Let $\mathcal{D} = \{\Omega_0, \dots, \Omega_{t-1}\}$ be a set of differential characteristics of a block cipher $E$. Then, $\mathcal{D}$ is called a singular cluster of $E$ if the corresponding effective keys form an empty set:*

$$\cap_{i=0}^{t-1} K_{\Omega_i} = \emptyset\,.$$

Here, a set of differential characteristics can be a differential, a truncated differential, or a multiple differential. It is common to compute the probability of the collection by summing up the probability of each characteristic in practice. However, if one shows that the collection is singular, the estimation might be more or less inaccurate. Note that the probability of a singular cluster is not necessarily zero, which is different from singular characteristics.

At a first glance, one can determine the effective keys of each differential characteristic separately, and then find the intersection of these effective keys. However, we have a more efficient algorithm for two characteristics: For 2-round differential characteristics $\Omega$ and $\Omega^*$, we write the effective keys in these characteristics as $\mathcal{K} = Ax \oplus B$ and $\mathcal{K}^* = A^* x^* \oplus B^*$, respectively. The effective keys $k \in \mathcal{K}, k^* \in \mathcal{K}^*$ satisfy $k = k^*$, which is linear with respect to $x$ and $x^*$. Therefore, if this linear system has no solutions, we can declare that $\Omega$ and $\Omega^*$ form a singular cluster. To determine the singularity of the equation systems is equivalent to compare the ranks of the coefficient matrix $[A, A^*]$ and the augmented matrix $[A, A^*, B \oplus B^*]$. The strategy can be generalised for any number of $r$-round characteristics, for simplicity, here we show it for a

pair of $r$-round characteristics in Algorithm 6. Notice that we only compare the effective keys of the characteristics in the same segment, however, it is also possible to find a singular characteristic by taking the key schedule into consideration, *i.e.*, exploring the effective keys in different segments of two characteristics.

**Remark 10.** *Here we note a difference between singular characteristics and singular clusters. A characteristic being singular implies that its probability is zero for all master keys. However, the probability of characteristics in a singular cluster is not necessarily zero, since the singular behaviour is that some of the characteristics in the cluster cannot exist simultaneously for every master key. Moreover, for a collection of characteristics, if there are two characteristics forming a singular cluster on their own, the collection is again a singular cluster.*

## 6.4 Differential Properties of the AES

### 6.4.1 Singular Characteristics of the AES

Since its publication, the security of the AES has been studied intensively. Moreover, because of the well-studied properties of the AES, other primitives such as hash functions also adopt a similar design strategy. For example, the SHA-3 candidate Grøstl [90] adopts two permutations which are similar to the AES. In the cryptanalysis of these primitives, differential characteristics with active pattern $1 \to m \to m^2 \to m \to 1$ are widely used where the states of these primitives are viewed as $m \times m$ square matrices over finite fields.[1] For example, as a newly developed technique in the meet-in-the-middle attack, the differential enumerating technique [72] is based on a 5-round differential characteristic of this pattern; in the rebound attack on Grøstl, a similar differential characteristic was adopted [156].

It is not difficult to find singular characteristics for the AES; for instance, the following differential characteristic of 5-round AES-128 is singular, where

[1] An active pattern indicates the number of active S-boxes in the corresponding round of the characteristic.

5-round AES-128 is defined as the first 5 rounds of the AES-128.

$$\Omega_1 : \begin{pmatrix} 1&0&0&0\\ 0&0&0&0\\ 0&0&0&0\\ 0&0&0&0 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 1&0&0&0\\ 0&0&0&0\\ 0&0&0&0\\ 0&0&0&0 \end{pmatrix} \xrightarrow{P} \begin{pmatrix} 2&0&0&0\\ 1&0&0&0\\ 1&0&0&0\\ 3&0&0&0 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 3&0&0&0\\ 1&0&0&0\\ 1&0&0&0\\ 2&0&0&0 \end{pmatrix} \xrightarrow{P} \begin{pmatrix} 6&2&1&3\\ 3&2&3&2\\ 3&6&2&1\\ 5&4&1&1 \end{pmatrix}$$

$$\xrightarrow{S} \begin{pmatrix} 24&27&39&9D\\ 45&36&36&27\\ 36&F1&2E&2D\\ 39&2D&1F&3A \end{pmatrix} \xrightarrow{P} \begin{pmatrix} 6&0&0&0\\ 0&5&0&0\\ 0&0&5&0\\ 0&0&0&36 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} E&0&0&0\\ 0&9&0&0\\ 0&0&D&0\\ 0&0&0&B \end{pmatrix} \xrightarrow{P} \begin{pmatrix} 1&0&0&0\\ 0&0&0&0\\ 0&0&0&0\\ 0&0&0&0 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 1&0&0&0\\ 0&0&0&0\\ 0&0&0&0\\ 0&0&0&0 \end{pmatrix}.$$

Here we note that a characteristic is singular if any of its segments is singular. For instance, the middle 3-round of $\Omega_1$ is singular, and $\Omega_1$ can be regarded as the segment of some characteristic covering more rounds which is singular as well.

There is no doubt that the singularity of differential characteristics is sensitive to the key schedule. For two ciphers with the same round function but different key schedules, it is likely to find characteristics which are valid for one but singular for another. Here, by encrypting a pair of messages with a 128-bit random master key through the AES, and tracking the difference propagation for 3 rounds (from round 3 to 5), we get a valid 3-round characteristic of the AES-128:[2]

$$\Omega_2 : \begin{pmatrix} C&AE&21&17\\ 8&57&21&39\\ 4&57&63&2E\\ 4&F9&42&17 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} D9&FD&94&7E\\ 15&D8&51&F2\\ F3&EE&14&7C\\ EC&EB&8B&B7 \end{pmatrix} \xrightarrow{P} \begin{pmatrix} 79&82&26&A6\\ F9&37&8E&F6\\ EB&7B&BD&2A\\ C9&F2&6B&74 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 5&24&B0&94\\ 7B&F4&FC&E8\\ 8&C&3E&A3\\ 8D&FE&9C&C3 \end{pmatrix}$$

$$\xrightarrow{P} \begin{pmatrix} F0&79&AE&2E\\ 77&B4&9D&EA\\ D3&9&51&48\\ 58&32&CC&F3 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 99&5E&74&5\\ 3E&23&AF&88\\ 5F&DD&49&7E\\ 19&60&95&AA \end{pmatrix}.$$

However, it can be verified that $\Omega_2$ is actually a singular characteristic of AES-192 (from round 3 to 5), which implies a valid attack based on differential characteristics on AES-128 might be invalid on AES-192. Here, we conjecture that in general a valid characteristic would probably turn into a singular one when the key schedule is modified.

[2] Although characteristics with fewer active S-boxes are often preferred by attackers, it is difficult to confirm that such characteristics are not singular. So we construct the example in such a way that the characteristic is guaranteed to possess at least two right inputs.

### 6.4.2 Density of Singular Characteristics in the AES

The existence of singular characteristics shows the hidden behaviour of characteristics interacting with the key schedule. With examples of singular characteristics found for the AES in previous sections, it also needs to be pointed out that the singular characteristics are far more common than just a coincidence.

**Definition 25.** *Let $\alpha \to \beta$ be a differential of a cipher and $\mathcal{D} = \{\Omega_0, \ldots, \Omega_{t-1}\}$ be all the differential characteristics in the differential. The density of $\alpha \to \beta$, denoted by $\rho(\alpha \to \beta)$, is defined as the fraction of singular characteristics in $\mathcal{D}$.*

As discussed earlier, a randomly picked characteristic is likely not valid in reality. In particular, we concentrate on the singularity of the characteristics in AES-128 with the pattern $1 \to 4 \to 16 \to 4 \to 1$ which are the majority of all characteristics constructed in the classical enumerating approach.

We carry out experiments to identify singular characteristics in 3-round differentials containing characteristics with the same active pattern as follows, where $*$ stands for a nonzero difference. Furthermore, we set the input and output differences such that the 3-round characteristic from the middle three rounds of a 5-round characteristic with active pattern $1 \to 4 \to 16 \to 4 \to 1$.

$$\begin{pmatrix} * & 0 & 0 & 0 \\ * & 0 & 0 & 0 \\ * & 0 & 0 & 0 \\ * & 0 & 0 & 0 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} * & 0 & 0 & 0 \\ * & 0 & 0 & 0 \\ * & 0 & 0 & 0 \\ * & 0 & 0 & 0 \end{pmatrix} \xrightarrow{P} \begin{pmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix} \xrightarrow{S} \begin{pmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix} \xrightarrow{P} \begin{pmatrix} * & 0 & 0 & 0 \\ 0 & * & 0 & 0 \\ 0 & 0 & * & 0 \\ 0 & 0 & 0 & * \end{pmatrix} \xrightarrow{S} \begin{pmatrix} * & 0 & 0 & 0 \\ 0 & * & 0 & 0 \\ 0 & 0 & * & 0 \\ 0 & 0 & 0 & * \end{pmatrix}.$$

By randomly choosing 1024 differentials $\mathcal{D}_0, \ldots, \mathcal{D}_{1023}$, we enumerate all the characteristics in each differential when the differences after the first S-box layer are fixed (but randomly chosen within the possible output differences), and check the singularity with the linear equation systems derived from the key schedule of AES-128. The average density of singular characteristics is approximately

$$\bar{\rho} = 98.47\%.$$

It is necessary to point out that the nonlinear operations of a key schedule are not explored in the linear equation systems, thus, the density of singular characteristics derived here is actually a lower bound. To be specific, when one adopts a certain automated search tool in finding such differential characteristics in the AES, less than 2% of the characteristics found by the tool actually exist in reality. It is not the failure of any tool, but the fact that the method of enumerating differential characteristics simply ignores the influence of the key schedules.

By looking into the linear equation systems, we are able to theoretically estimate the density of singular characteristics. Assume that the equation system $\mathbb{L}'(X) = 0$ of such a characteristic has $N$ equations and the rank of the coefficient matrix is $\tilde{R}$. Then the dimension of the image space for the coefficient matrix is $\tilde{R}$, which means the probability that a given characteristic is singular is no less than $1 - 2^{\tilde{R}-N}$, when the augment column is random. In the above experiments, the rank of the coefficient matrix in the linear equation system takes values from $\tilde{R} = 88$ to $\tilde{R} = 90$. Hence, in theory the density of singular characteristics is around $1 - 2^{-6}$ to $1 - 2^{-8}$ (*i.e.* 98.43% to 99.61%), which is consistent with the experiments. As a result, the assumption on the randomness of the augment columns is rather reasonable, which may attribute to the confusion effect of the S-boxes on the right inputs and right outputs. In other words, the augment column may take almost any values from the vector space, since there is no simple (linear) relation between the right pairs of two different difference propagations through the S-box.

Intuitively speaking, singular characteristics are more dense when the number of equations $N$ is considerably larger than the rank of the coefficient matrix $\tilde{R}$, which happens if the differences are of higher Hamming weights such that additional equations can be generated while the rank of the coefficient matrix is lower in general. For example, the density of 3-round characteristics with all differences being nonzero in AES-128 is estimated to be around $1 - 2^{-68}$ since the corresponding linear equation system has 96 equations and rank no larger than 28. Even though the total number of such 3-round characteristics can be tremendous, the number of valid characteristics may only be marginal which leads to an obvious effect on the estimation of the probability of a differential.

**Definition 26.** *A differential $\alpha \to \beta$ is singular if all its characteristics are singular, which is to say, $\rho(\alpha \to \beta) = 1$.*

A singular differential is in fact an impossible differential, however the former will be overlooked under previous techniques of finding impossible differentials. The large densities of the singular characteristics suggest that they may lead to a new method to constructing impossible differentials, *i.e.*, if all differential characteristics starting from $\alpha$ and ending at $\beta$ are detected to be singular, we can conclude that $\alpha \to \beta$ is an impossible differential. However, due to the huge number of differential characteristics within a differential, it may be very difficult to directly test all of them sequentially.

### 6.4.3 Singular Cluster in the AES

As an example, we find that the following 2-round differential characteristics of the AES-128 form a singular cluster:

$$\Omega_3 : \begin{pmatrix} 1\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 18\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \end{pmatrix} \xrightarrow{P} \begin{pmatrix} 30\,0\,0\,0 \\ 18\,0\,0\,0 \\ 18\,0\,0\,0 \\ 28\,0\,0\,0 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 3\,0\,0\,0 \\ 1\,0\,0\,0 \\ 1\,0\,0\,0 \\ 2\,0\,0\,0 \end{pmatrix},$$

$$\Omega_4 : \begin{pmatrix} 1\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 14\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \end{pmatrix} \xrightarrow{P} \begin{pmatrix} 28\,0\,0\,0 \\ 14\,0\,0\,0 \\ 14\,0\,0\,0 \\ 3\mathtt{C}\,0\,0\,0 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 3\,0\,0\,0 \\ 1\,0\,0\,0 \\ 1\,0\,0\,0 \\ 2\,0\,0\,0 \end{pmatrix}.$$

**Remark 11.** *Singular clusters are not rare. Recall that the singularity can be identified by a linear equation system: we expect a similar behaviour as the characteristics being singular, that is, the linear equation system derived for a set of characteristics is more likely to be singular if differences in the characteristics are of higher Hamming weights. The singular characteristics and singular clusters can be detected for many lightweight designs, especially when they adopt an AES-like structure, this presents a concern for the attackers with respect to the effectiveness of cryptanalysis based on differential characteristics. As an illustration, we will show the singular characteristics and singular clusters of the* PRINCE *cipher in next section; these results have a practical influence on previous studies such as multiple differential cryptanalysis [56].*

## 6.5 A Tale of Two Perspectives

An important observation in previous sections, is that the differential characteristics have a much closer connection with the keys than the statement in the hypothesis of stochastic equivalence. The existence of singular characteristics and singular clusters shows that the information of the keys and the key schedules should be taken into consideration, not merely for related-key models and other open-key models as we previously believed.

Consider the following illustrative example. Assume that $\Omega$ is a valid 10-round differential characteristic with at least one pair of right inputs in the AES-128. We can find such a characteristic by encrypting a pair of plaintexts under a random key, and the solutions to the linear equation system are candidates for the effective keys. In most experiments we carried out, the only candidate

from the solution of the equation system is exactly the master key used in the encryption.

What we believe critical is that the hypothesis of stochastic equivalence and Markov model need to be interpreted from two different perspectives: the designer and the attacker.

**Designer's Perspective.** When designing a block cipher or a permutation, the model based on the hypothesis of stochastic equivalence provides an approximate view of an overall behaviour possessed by general characteristics under all independently chosen random keys. Under such a scenario, it makes sense that the designers take the security bound as a main factor of consideration rather than a particular characteristic having probability 0 or a marginal nonzero probability. For instance, it is proven that there are at least 25 active S-boxes in any 4-round differential characteristic of the AES such as the active pattern of $\Omega_1$ in Section 6.4; however, the provable bound might not be guaranteed to be tight for certain ciphers.

**Attacker's Perspective.** For the attackers whose main target is to identify one specific non-randomness in the primitives, the hypothesis of stochastic equivalence for a designer's perspective might lead to a distorted image.[3] One of the underestimated factors is the role of the key schedule as we show for singular characteristics, which is unfortunately often ignored in many practical analyses.

PRINCE is an AES-like primitive, hence it possesses singular characteristics and singular clusters as well. The following 3-round characteristic of PRINCE is singular even though its EDP is as high as $2^{-35}$.

$$\Omega_5 : \begin{pmatrix} 8&0&4&0\\0&0&0&0\\4&0&8&0\\0&0&0&0 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 8&0&4&0\\0&0&0&0\\8&0&4&0\\0&0&0&0 \end{pmatrix} \xrightarrow{M'} \begin{pmatrix} 8&0&4&0\\0&0&0&0\\8&0&4&0\\0&0&0&0 \end{pmatrix} \xrightarrow{SR} \begin{pmatrix} 8&0&4&0\\0&0&0&0\\4&0&8&0\\0&0&0&0 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 8&0&5&0\\0&0&0&0\\8&0&5&0\\0&0&0&0 \end{pmatrix}$$

$$\xrightarrow{M'} \begin{pmatrix} 8&0&5&0\\0&0&0&0\\8&0&5&0\\0&0&0&0 \end{pmatrix} \xrightarrow{SR} \begin{pmatrix} 8&0&5&0\\0&0&0&0\\5&0&8&0\\0&0&0&0 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 2&0&5&0\\0&0&0&0\\2&0&5&0\\0&0&0&0 \end{pmatrix}$$

It enlightens us that an invalid differential characteristic can be easily mistaken as real if one neglects the impact of key schedules.

[3]Some experiments show that the estimation under the hypothesis is rather close to reality, see for instance, [130] It is noteworthy that the irregularity is what the attackers have to pay extra attention to, which is supported by a number of studies such as those we have previously referred to in this chapter.

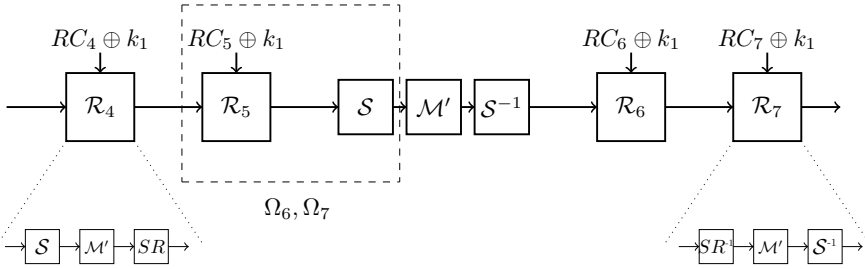



Figure 6.1: PRINCE$_{core}$ reduced to 6 rounds. The dashed box shows the location of $\Omega_6$ and $\Omega_7$ in the 6-round characteristics adopted for a multiple differentials distinguisher.

Moreover, multiple differential cryptanalysis has been applied to PRINCE [56]. The multiple differentials studied in PRINCE consist of characteristics with active patterns as $\Omega_5$. The cipher is reduced to 6 rounds of PRINCE$_{core}$ which is illustrated in Figure 6.1. The input and output differences are $(\delta_{in}, \delta_{out}) \in \{(1,2),(2,1)\} \times \{(1,2),(2,1)\}$. We found the following singular cluster in PRINCE.

$$\Omega_6 : \begin{pmatrix} 8&0&4&0\\0&0&0&0\\4&0&8&0\\0&0&0&0 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 5&0&8&0\\0&0&0&0\\5&0&8&0\\0&0&0&0 \end{pmatrix} \xrightarrow{M'} \begin{pmatrix} 0&0&0&0\\5&0&8&0\\0&0&0&0\\5&0&8&0 \end{pmatrix} \xrightarrow{SR} \begin{pmatrix} 0&0&0&0\\0&8&0&5\\0&0&0&0\\0&5&0&8 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 0&0&0&0\\0&5&0&2\\0&0&0&0\\0&5&0&2 \end{pmatrix},$$

$$\Omega_7 : \begin{pmatrix} 8&0&1&0\\0&0&0&0\\1&0&8&0\\0&0&0&0 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 4&0&8&0\\0&0&0&0\\4&0&8&0\\0&0&0&0 \end{pmatrix} \xrightarrow{M'} \begin{pmatrix} 0&0&0&0\\4&0&8&0\\0&0&0&0\\4&0&8&0 \end{pmatrix} \xrightarrow{SR} \begin{pmatrix} 0&0&0&0\\0&8&0&4\\0&0&0&0\\0&4&0&8 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 0&0&0&0\\0&5&0&2\\0&0&0&0\\0&5&0&2 \end{pmatrix}.$$

These are the fragments of two 6-round characteristics with input difference

$$\begin{pmatrix} 2&0&1&0\\0&0&0&0\\2&0&1&0\\0&0&0&0 \end{pmatrix},$$

which are located in the dashed box as depicted in Figure 6.1.

Characteristics forming a singular cluster indicates that at most one of them is valid under all possible keys, such that they show no congregating effect if multiple differentials are considered. Furthermore, since round-reduced PRINCE is symmetric with respect to the middle round, the characteristics $\Omega_6, \Omega_7$ in reversed order are still within a singular cluster located in the last few rounds.

As a matter of fact, the reflection property of the PRINCE cipher can be further utilised in order to construct singular characteristics. For instance, connecting $\Omega_6$ and $\Omega_7$ in their tails by the middle switch $\mathcal{M}'$, we get a 4-round characteristic $\Omega_8$ as below. Taking the constants into account, it can be verified that $\Omega_8$ is indeed a singular characteristic covering the middle 4 rounds as in Figure 6.1, which was considered as valid to build a distinguisher [56].

$$\Omega_8 : \begin{pmatrix} 8&0&4&0\\0&0&0&0\\4&0&8&0\\0&0&0&0 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 5&0&8&0\\0&0&0&0\\5&0&8&0\\0&0&0&0 \end{pmatrix} \xrightarrow{M'} \begin{pmatrix} 0&0&0&0\\5&0&8&0\\0&0&0&0\\5&0&8&0 \end{pmatrix} \xrightarrow{SR} \begin{pmatrix} 0&0&0&0\\0&8&0&5\\0&0&0&0\\0&5&0&8 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 0&0&0&0\\0&5&0&2\\0&0&0&0\\0&5&0&2 \end{pmatrix}$$

$$\xrightarrow{M'} \begin{pmatrix} 0&0&0&0\\0&5&0&2\\0&0&0&0\\0&5&0&2 \end{pmatrix} \xrightarrow{S^{-1}} \begin{pmatrix} 0&0&0&0\\0&8&0&4\\0&0&0&0\\0&4&0&8 \end{pmatrix} \xrightarrow{SR^{-1}} \begin{pmatrix} 0&0&0&0\\4&0&8&0\\0&0&0&0\\4&0&8&0 \end{pmatrix} \xrightarrow{M'} \begin{pmatrix} 4&0&8&0\\0&0&0&0\\4&0&8&0\\0&0&0&0 \end{pmatrix} \xrightarrow{S^{-1}} \begin{pmatrix} 8&0&1&0\\0&0&0&0\\1&0&8&0\\0&0&0&0 \end{pmatrix}$$

Note that the singular characteristics are sensitive to the key as well as the constant in this example. Our result shows that $\Omega_8$ is singular when the constants $RC_i$ and $RC_{i+1}$ XORed to the 4-round characteristic satisfy $i = 3, 4, 5, 6, 7$. The round constants can be found in the PRINCE specification [47]. Therefore, the existence of singular characteristics and singular clusters shows that the cryptanalytic results of PRINCE in [56] need to be scrutinised.

## 6.6 Concluding Remarks

The key schedule of a block cipher is rarely exploited in differential cryptanalysis. The hypothesis of stochastic equivalence and the Markov model are the foundations to evaluate the security of block ciphers against differential cryptanalysis and serve as a guideline for design cryptographic primitives. In this chapter, we show that it may lead to incorrect results when applying characteristic-based differential cryptanalysis to a real cipher without taking the key schedule into account.

To show our point of view, we propose the concept of singular characteristics by studying their effective keys, *i.e.*, characteristics with nonzero EDP but with probability 0 for all master keys. In addition, we study the congregating effect of characteristics and propose singular clusters to find multiple differential characteristics with no effective keys. We found examples of a 5-round singular characteristic in the AES, by investigating the property of its key schedule, and showed that the singular characteristics in differentials of the AES are in

fact the overwhelming majority with density close to 1. We also construct a valid differential characteristic of AES-128 while it is proved to be singular in AES-192 to demonstrate that the existence of a differential characteristic is sensitive to the key schedule.

It is also interesting to note that the opposite of singular characteristics may also be useful for attackers. If a characteristic has high probability for most of the genuine keys, it is in fact a good distinguisher even though the EDP of the characteristic might be marginal. A similar phenomenon has been observed through the analysis of PRINTcipher in multidimensional linear cryptanalysis [136, Section 4.3].

# Chapter 7

# Conclusion

In this thesis, we focused on the cryptanalysis of symmetric-key primitives, especially block ciphers. Design and analysis are the core aspects of symmetric-key cryptography, where the goal in common is to improve the security of the primitives. Even though the motivation of the thesis leans towards finding improved distinguishers and attacks on specific ciphers, we do hope to provide insights and solutions for future designs as well, since a good design always keeps the attacks in mind.

One of the main contributions in this thesis is the automatic search of distinguishers with the applications to cryptanalysis and design in Chapter 2, especially for the ARX structure. The automatic search techniques based on a certain solver have been successfully applied to finding lower bounds on the number of active S-boxes in SPN ciphers for several years. However, the research has only gradually developed during the past few years for the ciphers with Boolean operations and arithmetic, partially motivated by the publication of the NSA ciphers Simon and Speck. One major obstacle to be overcome back then was to find a solver-friendly mathematical description of the distinguishers, such as the propagation of the differences or linear masks through the round function. This problem motivates the first part of this thesis. With a special focus on the ARX ciphers, a novel cryptanalytic technique called rotational-XOR cryptanalysis is proposed, which is based on a statistical distinguisher combining both differential cryptanalysis and rotational cryptanalysis. In addition, automatic search techniques based on SAT/SMT solvers are developed with applications to the linear cryptanalysis and rotational-XOR cryptanalysis of ARX ciphers. A Python-based tool ArxPy accelerates the analysis process by parsing a cipher and writing the SAT/SMT files automatically. As an

application, improved distinguishers are found for the block cipher family Speck. While solver-based automatic search techniques have made a notable progress in analysing ARX ciphers, they have also been adopted in SPN ciphers for searching the characteristics and trails, where the modelling of the S-boxes (even 8-bit ones) becomes feasible. There is no doubt that automatic search techniques have replaced some efforts in cryptanalysis which used to be carried out with a large amount of human hours.

Meanwhile, new symmetric primitives have been proposed with an increasing speed over decades, motivated by the IoT application scenarios. As we have recalled in the first chapter, new block ciphers as well as lightweight hash functions and AE schemes have been designed to optimise the performance under software and hardware constraints. Researchers have adopted novel structures and components in order to push the designs to the limit. In addition, ciphers with resistance against side-channel attacks receive attention from the theoretical and practical viewpoints. Interestingly, one reason behind the proliferation of designs is the development in the area of automatic search, which efficiently assists the designers in manipulating the trade-offs between security and performance.

During the past years, questions have frequently been asked *whether new types of cryptanalysis can be found* and *if better ciphers can be designed.*

Indeed, many cryptanalytic methods were invented in the 1990's, while a number of symmetric primitives have been standardised by international and national organisations for a wide spectrum of application scenarios. Nevertheless, the pace of symmetric-key cryptography never slows down. To begin with, invariant subspace attack, the division property, the nonlinear invariant attack, and also rotational-XOR cryptanalysis discussed in this thesis are just a fraction of the novel analysis techniques we have seen in recent years, some of which are developed in the context of lightweight primitives. In addition, new techniques are proposed to improve the attacks as well, to name a few, the first practical collision of full SHA-1 was published in 2017 by a group of researchers from CWI and Google research [184]; integral cryptanalysis with improved key recovery techniques successfully broke the block cipher Misty [192]; the applications of the linear structures technique by Guo *et al.* lead to improved pre-image and collision attacks in round-reduced Keccak [100]. Another example is the optimised interpolation attacks on the block cipher LowMC in Chapter 3. As a cipher aiming at a novel feature, the designers of LowMC introduced many new ideas and attempts on minimising the multiplicative complexity. However, our study has shown that the security margin of the (first version of the) cipher is not enough to resist higher-order differential cryptanalysis and the interpolation attack. In addition, several new designs aiming at low multiplicative complexity were proposed over the last few years, some of them chose to use fewer nonlinear

operations per round and extended the total number of rounds, such as MiMC. Further applications of the improved interpolation attack on those ciphers would be an interesting topic.

With the selection of Keccak as the SHA-3 standard, permutation-based cryptography was proposed by the KeccakTeam [28]; they showed that most mainstream types of symmetric primitives can be efficiently designed with permutations with a sponge construction. Chapter 6 of this thesis studies differential cryptanalysis against fixed-key block ciphers, where the observation may be beneficial for permutations as well. Our study shows that the application of differential cryptanalysis in fixed-key block ciphers may result in a different picture than the one predicted by the basic assumptions such as the hypothesis of stochastic equivalence, especially for the characteristics. We focus on the characteristics with nonzero expected differential probabilities, which are actually of zero probability for all master keys. An efficient algorithm is developed in our research, with an application to 5-round differential characteristics in the AES. Our findings are of interest for permutations, because the round constants in permutations are similar to a "known" fixed key. If a differential can be shown to be in fact an impossible one under a specific combination of the round constants, then a distinguisher is found.

This thesis focused on several interesting problems in design as well. In terms of new designs, they are expected to offer resistance against known cryptanalytic methods, especially with features to demonstrate such resistance with mathematical deductions, such as the Wide Trail Strategy. In Chapter 4, we study the resistance against invariant subspace attacks for some AES-like ciphers, by considering the properties of the linear layer. Based on a condition on the linear layer, it can be shown that the dimension of possible invariant subspaces can be bounded, which leads to a countermeasure in choosing the constants to break the iterative propagation of the invariant subspaces in a cipher.

In recent years, a large fraction of new block ciphers choose to take the SPN structure, where the main design focus is on 4-bit S-boxes and lightweight linear layers with MDS or NMDS property. Notably, a series of studies try to construct MDS or NMDS matrices with smaller XOR count [20, 142], some of which are adopted in new lightweight designs. Instead of searching for new linear functions, Chapter 5 proposes a novel component called the nonlinear diffusion layer. We argue that nonlinear diffusion layers are beneficial for both diffusion and confusion effects in a block cipher, which could guarantee the resistance against differential and linear cryptanalysis within a smaller number of rounds. Two types of nonlinear diffusion functions are proposed, with their cryptographic properties analysed. It is promising for designers to adopt the nonlinear diffusion functions we proposed, or to construct new nonlinear layers

for a new cipher, based on the observations from the toy ciphers we studied.

Our results as summarised in this thesis are only a tiny fraction of the overall efforts and ideas from the researchers in the community which, in total, provide positive solutions to partially respond to the questions raised earlier.

One important future work is to improve the automatic search techniques in several aspects. For instance, one technical barrier of the solver-based automatic search lies in the exponential growth of the complexity with respect to the block size and the number of rounds. A more efficient method to translate the problems for the solvers could be advantageous, while it is also promising to integrate the cryptographic problems into specialised solvers such as Cryptominisat. Meanwhile, the analysis of block ciphers is largely automated, but the design still requires the expertise of the researchers. From the viewpoint of this thesis, one of the most promising attempts would be an ARX cipher with automatic dynamic design and analysis, for the simple structure and operations in ARX. Speaking of automated techniques and algorithms, it would be interesting to utilise machine learning algorithms as an alternative to the solvers. On the other hand, it is feasible to search for characteristics in SPN ciphers as well, but the effect of the singular characteristics is often not taken into account. Therefore, it is possible that many of the characteristics found by a solver are in fact invalid in practice. It is interesting to combine the conditions on singular characteristics with automatic search techniques, which avoids invalid characteristics and narrows down the search space.

Another exciting research topic is the nonlinear diffusion layer, especially the construction of new nonlinear diffusion functions and the design of ciphers with a nonlinear diffusion layer. Considering the ARX structure in one of the nonlinear functions we proposed, it is certainly interesting to take both the Wide Trail Strategy (which mainly focuses on SPN ciphers) and the Long Trail Strategy [73] (which focuses on ARX ciphers) into account, in order to find new nonlinear diffusion functions based on Boolean arithmetic and operations.

# Appendix A

# Mathematical Background

Many cryptographic primitives are constructed based on mathematical structures. In symmetric-key cryptography, one of the most important mathematical objects is the Boolean function. The research on Boolean functions is an active topic in the mathematical aspects of information theory [57]. Here we recall some basics about finite fields and Boolean functions.

**Definition 27.** *Let $G$ be a set, $+$ be an operation. We call $(G,+)$ a group if (1) $\forall g_1, g_2 \in G$, $g_1+g_2 \in G$; (2) $\forall g_1, g_2, g_3 \in G$, $(g_1+g_2)+g_3 = g_1+(g_2+g_3)$; (3) there exists an element $0$, such that $\forall g \in G$, $g+0=g$; (4) $\forall g \in G$, there exists an element $g'$, such that $g+g'=0$. Furthermore, if $\forall g_1, g_2 \in G$, $g_1+g_2 = g_2+g_1$, then $G$ is an Abelian group.*

**Definition 28.** *Let $R$ be a set, $+, \times$ be two (different) operations. We call $(R,+,\times)$ a ring if (1) $(R,+)$ is an Abelian group; (2) $\forall r_1, r_2, r_3$, $(r_1 \times r_2) \times r_3 = r_1 \times (r_2 \times r_3)$; (3) $\forall r_1, r_2, r_3 \in R$, $r_1 \times (r_2 + r_3) = r_1 \times r_2 + r_1 \times r_3$, $(r_1+r_2) \times r_3 = r_1 \times r_3 + r_2 \times r_3$.*

**Definition 29.** *Let $p$ be a prime, $\mathbb{F}_p$ is the prime field with the elements $\{0,1,2,\ldots,p-1\}$. Assume that $f(x)$ is a irreducible polynomial (i.e., a polynomial with no nontrivial divisors) of degree $n$ in the polynomial ring $\mathbb{F}_p[x]$, $<f(x)>$ is the ring with all the polynomials which have $f(x)$ as a divisor. Then, the residue field $\mathbb{F}_p[x]/<f(x)>$ is a finite field of $p^n$ elements, which is denoted by $\mathbb{F}_{p^n}$.*

**Proposition 5.** *Assume that $F$ is a finite field, $+, \times$ are the operations. Then $(F,+)$ is an Abelian group, $(F\backslash\{0\}, \times)$ is an Abelian group.*

Finite fields lies at the mathematical intersection of computer science and communication technology, where the most well-known example is the binary field. It contains only two elements $\{0,1\}$, *i.e.*, the on and off states in circuits. Boolean functions are functions over the binary field mapping $\mathbb{F}_{2^n}$ to $\mathbb{F}_2$. The algebraic normal form (ANF) of a Boolean function with degree $n$ is defined as

$$\bigoplus_{I\in\mathcal{P}(N)} a_I(\prod_{i\in I} x_i) = \bigoplus_{I\in\mathcal{P}(N)} a_I x^I\,,$$

where $\mathcal{P}(N)$ is the power set of $\{1,2,\ldots,n\}$. The algebraic degree is the maximum degree of the monomials in the ANF, *i.e.*, $\max\{|I||a_I \neq 0\}$. The total number of Boolean functions of $n$ variables is $2^{2^n}$. In many applications, permutations over finite fields, or vectorial Boolean functions are studied. A vectorial Boolean function $\hat{f} : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ consists of $n$ Boolean functions $f_i : \mathbb{F}_{2^n} \to \mathbb{F}_2$, which are called the component functions. The algebraic degree of a vectorial Boolean function is defined as the global degree $\max\{\deg(f_i)\}$.

In order to gain an systematical view instead of one specific function, equivalence classes can be defined for Boolean functions and vectorial Boolean functions. Here, we recall the affine equivalence and CCZ equivalence.

**Definition 30** ([57])**.** *Let $f$ and $g$ be Boolean functions. $f$ and $g$ are called affine equivalent if there exists a linear functions $L$ and a element $a$, such that for all $x$, we have $f(x) = g(L(x) + a)$.*

**Definition 31** ([57])**.** *Let $f$ and $g$ be Boolean functions. $f$ and $g$ are called CCZ equivalent if their graphs $C_f = \{(x, y)|y = f(x)\}$ and $C_g = \{(x, y)|y = g(x)\}$ are affine equivalent.*

Boolean functions with affine equivalence are of the same algebraic degree and share many similar cryptographic properties. Compared with affine equivalence, CCZ equivalence, named after three researchers who discovered the property, is more general. The degree of two CCZ-equivalent functions may differ.

The security of block ciphers depends on the properties of the building blocks, one important factor is the S-box. In order to evaluate the resistance of the S-boxes against differential and linear cryptanalysis, differential uniformity and nonlinearity are studied.

**Definition 32.** *Let $\hat{f}$ be a vectorial Boolean function from $\mathbb{F}_{2^n}$ to $\mathbb{F}_{2^m}$. The differential uniformity is defined as*

$$\max_{a\in\mathbb{F}_{2^n}, b\in\mathbb{F}_{2^m}} |\{x|\hat{f}(x \oplus a) \oplus \hat{f}(x) = b\}|.$$

*The nonlinearity is the distance between the Boolean function and all linear functions, which is defined as follows:*

$$2^{n-1} - \frac{1}{2} \max_{v \in \mathbb{F}_{2^m}^*, u \in \mathbb{F}_{2^n}} | \sum_{x \in \mathbb{F}_{2^n}} (-1)^{u \cdot x \oplus v \cdot \hat{f}(x)} |.$$

Vectorial Boolean functions with optimal differential uniformity (which equals 2) are called almost perfect nonlinear (APN). At the time of writing this thesis, APN functions with $n$ being even have only been found up to $n = 6$ [51]. Note that the 8-bit S-box of the AES is differentially 4-uniform.

For ciphers without S-boxes, it is interesting to study the nonlinear operations such as the biswise AND and the modular addition. For instance, when analysing the cryptographic properties of modular addition, we often consider it as a T-function or an S-function.

**Definition 33** ([120])**.** *A function $f : \mathbb{F}_{2^n}^l \to \mathbb{F}_{2^n}$ is called a T-function if the $i$-th bit of the output only depends on the $j$-th bits of the inputs where $j \leq i$.*

**Definition 34** ([160])**.** *An S-function accepts $n$-bit words $a_1, a_2, \cdots, a_k$ and a list of states $S[i]$ (for $0 \leq i < n$) as inputs, and produces an $n$-bit output word $b$ in the following way:*

$$(b[i], S[i+1]) = f(a_1[i], a_2[i], ..., a_k[i], S[i]), 0 \leq i < n.$$

Obviously, S-functions are in fact T-functions under the condition that the dependence between the output and input bits should be described by a finite state. In this thesis, we refer to both of them as T-functions.

# Bibliography


[1] Al Fardan, N. J., and Paterson, K. G. Lucky thirteen: Breaking the TLS and DTLS record protocols. In *IEEE Symposium on Security and Privacy - IEEE S&P 2013* (2013), IEEE, pp. 526–540.

[2] Albrecht, M., Grassi, L., Rechberger, C., Roy, A., and Tiessen, T. MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In *Advances in Cryptology - ASIACRYPT 2016* (2016), Springer, pp. 191–219.

[3] Albrecht, M., Rechberger, C., Schneider, T., Tiessen, T., and Zohner, M. Ciphers for MPC and FHE. In *Advances in Cryptology - EUROCRYPT 2015* (2015), Springer, pp. 430–454.

[4] Albrecht, M., Rechberger, C., Schneider, T., Tiessen, T., and Zohner, M. Ciphers for MPC and FHE. `https://eprint.iacr.org/2016/687`, 2016.

[5] Albrecht, M. R., Driessen, B., Kavun, E. B., Leander, G., Paar, C., and Yalçın, T. Block ciphers - focus on the linear layer (feat. PRIDE). In *Advances in Cryptology - CRYPTO 2014* (2014), Springer, pp. 57–76.

[6] Ashur, T., Beyne, T., and Rijmen, V. Revisiting the wrong-key-randomization hypothesis. `http://eprint.iacr.org/2016/990`, 2016.

[7] Ashur, T., and Liu, Y. Rotational cryptanalysis in the presence of constants. *IACR Transactions on Symmetric Cryptology 2016*, 1 (2016), 57–70.

[8] Aumasson, J.-P., Henzen, L., Meier, W., and Naya-Plasencia, M. Quark: A lightweight hash. *Journal of cryptology 26*, 2 (2013), 313–339.

[9] Aumasson, J.-P., Henzen, L., Meier, W., and Phan, R. C.-W. SHA-3 proposal BLAKE. *Submission to NIST* (2008).

[10] Aumasson, J.-P., Jovanovic, P., and Neves, S. NORX: parallel and scalable AEAD. In *European Symposium on Research in Computer Security* (2014), Springer, pp. 19–36.

[11] Aumasson, J.-P., and Meier, W. Zero-sum distinguishers for reduced Keccak-$f$ and for the core functions of Luffa and Hamsi. `http://131002.net/data/papers/AM09.pdf`, 2009.

[12] Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., and Regazzoni, F. Midori: a block cipher for low energy. In *Advances in Cryptology - ASIACRYPT 2015* (2015), Springer, pp. 411–436.

[13] Banik, S., Pandey, S. K., Peyrin, T., Sasaki, Y., Sim, S. M., and Todo, Y. GIFT: a small PRESENT. In *International Conference on Cryptographic Hardware and Embedded Systems - CHES 2017* (2017), Springer, pp. 321–345.

[14] Barker, E., and Roginsky, A. Transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths. *NIST Special Publication 800* (2015), 131A.

[15] Barrett, C., Fontaine, P., and Tinelli, C. The satisfiability modulo theories library SMT-LIB. `www.SMT-LIB.org`, 2016.

[16] Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., and Wingers, L. The SIMON and SPECK lightweight block ciphers. In *Proceedings of the 52nd Annual Design Automation Conference - DAC 2015* (2015), ACM, pp. 175:1–175:6.

[17] Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., and Wingers, L. Notes on the design and analysis of SIMON and SPECK. `http://eprint.iacr.org/2017/560`, 2017.

[18] Beierle, C., Canteaut, A., Leander, G., and Rotella, Y. Proving resistance against invariant attacks: How to choose the round constants. In *Advances in Cryptology - CRYPTO 2017* (2017), pp. 647–678.

[19] Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., and Sim, S. M. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *Advances in Cryptology - CRYPTO 2016* (2016), Springer, pp. 123–153.

[20] Beierle, C., Kranz, T., and Leander, G. Lightweight multiplication in GF($2^n$) with applications to MDS matrices. In *Advances in Cryptology - CRYPTO 2016* (2016), Springer, pp. 625–653.

[21] Bellare, M., Canetti, R., and Krawczyk, H. Keying hash functions for message authentication. In *Advances in Cryptology - CRYPTO'96* (1996), Springer, pp. 1–15.

[22] Bellare, M., Ristenpart, T., Rogaway, P., and Stegers, T. Format-preserving encryption. In *International Workshop on Selected Areas in Cryptography - SAC 2009* (2009), Springer, pp. 295–312.

[23] Bernstein, D. J. ChaCha, a variant of Salsa20. `http://cr.yp.to/chacha.html`.

[24] Bernstein, D. J. The Salsa20 family of stream ciphers. In *New stream cipher designs*. Springer, 2008, pp. 84–97.

[25] Bernstein, D. J. Introduction to post-quantum cryptography. In *Post-quantum cryptography*. Springer, 2009, pp. 1–14.

[26] Bertoni, G., Daemen, J., Hoffert, S., Peeters, M., Assche, G. V., and Keer, R. V. Farfalle: parallel permutation-based cryptography. Cryptology ePrint Archive, Report 2016/1188.

[27] Bertoni, G., Daemen, J., Peeters, M., Assche, G. V., and Keer, R. V. KangarooTwelve: fast hashing based on Keccak-p. Cryptology ePrint Archive, Report 2016/770.

[28] Bertoni, G., Daemen, J., Peeters, M., and Van Assche, G. Permutation-based encryption, authentication and authenticated encryption.

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

[30] Bertoni, G., Daemen, J., Peeters, M., and Van Assche, G. Keccak. In *Advance in Cryptology - EUROCRYPT 2013* (2013), Springer, pp. 313–314.

[31] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., and Van Keer, R. Ketje. *Submission to CAESAR* (2014).

[32] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., and Van Keer, R. Keyak. *Submission to CAESAR* (2014).

[33] Bhargavan, K., and Leurent, G. On the practical (in-)security of 64-bit block ciphers: Collision attacks on HTTP over TLS and OpenVPN. In *ACM SIGSAC Conference on Computer and Communications Security - CCS 2016* (2016), ACM, pp. 456–467.

[34] Biham, E., Biryukov, A., and Shamir, A. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In *Advances in Cryptology - EUROCRYPT '99* (1999), Springer, pp. 12–23.

[35] Biham, E., Dunkelman, O., and Keller, N. The rectangle attack - rectangling the Serpent. In *Advances in Cryptology - EUROCRYPT 2001* (2001), Springer, pp. 340–357.

[36] Biham, E., and Shamir, A. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology 4*, 1 (1991), 3–72.

[37] Biryukov, A., and Khovratovich, D. Related-key cryptanalysis of the full AES-192 and AES-256. In *Advances in Cryptology - ASIACRYPT 2009* (2009), pp. 1–18.

[38] Biryukov, A., Leurent, G., and Perrin, L. Cryptanalysis of Feistel networks with secret round functions. In *International Conference on Selected Areas in Cryptography - SAC 2015* (2015), Springer, pp. 102–121.

[39] Biryukov, A., and Perrin, L. State of the art in lightweight symmetric cryptography. `https://eprint.iacr.org/2017/511`, 2017.

[40] Biryukov, A., and Velichkov, V. Automatic search for differential trails in ARX ciphers. In *Topics in Cryptology - CT-RSA 2014* (2014), Springer, pp. 227–250.

[41] Biryukov, A., Velichkov, V., and Le Corre, Y. Automatic search for the best trails in ARX: Application to block cipher SPECK. In *International Workshop on Fast Software Encryption - FSE 2016* (2016), Springer, pp. 289–310.

[42] Bogdanov, A. *Analysis and design of block cipher constructions*. PhD thesis, Ruhr-Universität Bochum, 2009.

[43] Bogdanov, A., Knežević, M., Leander, G., Toz, D., Varici, K., and Verbauwhede, I. SPONGENT: A lightweight hash function. In *International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2011* (2011), Springer, pp. 312–325.

[44] Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J., Seurin, Y., and Vikkelsoe, C. PRESENT: An ultra-lightweight block cipher. In *International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2007* (2007), Springer, pp. 450–466.

[45] Bogdanov, A., and Rijmen, V. Linear hulls with correlation zero and linear cryptanalysis of block ciphers. *Designs, codes and cryptography 70*, 3 (2014), 369–383.

[46] Bogdanov, A., and Tischhauser, E. On the wrong key randomisation and key equivalence hypotheses in Matsui's Algorithm 2. In *International Workshop on Fast Software Encryption - FSE 2013* (2013), Springer, pp. 19–38.

[47] Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E. B., Knezevic, M., Knudsen, L. R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S. S., and Yalçin, T. PRINCE: a low-latency block cipher for pervasive computing applications. In *Advances in Cryptology - ASIACRYPT 2012* (2012), Springer, pp. 208–225.

[48] Borghoff, J., Knudsen, L. R., and Stolpe, M. Bivium as a mixed-integer linear programming problem. In *IMA Int. Conf.* (2009), Springer, pp. 133–152.

[49] Bouillaguet, C., Derbez, P., and Fouque, P.-A. Automatic search of attacks on round-reduced AES and applications. In *Advances in Cryptology - CRYPTO 2011* (2011), Springer, pp. 169–187.

[50] Boura, C., and Canteaut, A. A new criterion for avoiding the propagation of linear relations through an Sbox. In *International Workshop on Fast Software Encryption - FSE 2013* (2013), Springer, pp. 585–604.

[51] Browning, K., Dillon, J., McQuistan, M., and Wolfe, A. An APN permutation in dimension six. *Finite Fields: theory and applications 518* (2010), 33–42.

[52] Buchberger, B., and Winkler, F. *Gröbner bases and applications*, vol. 251. Cambridge University Press, 1998.

[53] Bulygin, S., Walter, M., and Buchmann, J. Full analysis of PRINTcipher with respect to invariant subspace attack: Efficient key recovery and countermeasures. *Designs, Codes and Cryptography 73*, 3 (2014), 997–1022.

[54] CAESAR. Competition for authenticated encryption: Security, applicability, and robustness. `https://competitions.cr.yp.to/caesar.html`, 2014.

[55] Canteaut, A., Carpov, S., Fontaine, C., Lepoint, T., Naya-Plasencia, M., Paillier, P., and Sirdey, R. Stream ciphers: A

practical solution for efficient homomorphic-ciphertext compression. In *International Conference on Fast Software Encryption - FSE 2016* (2016), Springer, pp. 313–333.

[56] Canteaut, A., Fuhr, T., Gilbert, H., Naya-Plasencia, M., and Reinhard, J. Multiple differential cryptanalysis of round-reduced PRINCE. In *International Workshop on Fast Software Encryption - FSE 2014* (2014), pp. 591–610.

[57] Carlet, C. Boolean functions for cryptography and error correcting codes. *Boolean models and methods in Mathematics, Computer science, and Engineering 2* (2010), 257.

[58] Chabaud, F., and Vaudenay, S. Links between differential and linear cryptanalysis. In *Advances in Cryptology – EUROCRYPT'94* (1994), Springer, pp. 356–365.

[59] Cook, S. A. The complexity of theorem-proving procedures. In *The third annual ACM symposium on Theory of computing* (1971), ACM, pp. 151–158.

[60] Courtois, N., and Pieprzyk, J. Cryptanalysis of block ciphers with overdefined systems of equations. In *Advances in Cryptology - ASIACRYPT 2002* (2002), Springer, pp. 267–287.

[61] Daemen, J. *Cipher and hash function design, strategies based on linear and differential cryptanalysis.* PhD thesis, KU Leuven, 1995.

[62] Daemen, J., Govaerts, R., and Vandewalle, J. Correlation matrices. In *International Workshop on Fast Software Encryption - FSE'95* (1995), Springer, pp. 275–285.

[63] Daemen, J., and Rijmen, V. The wide trail design strategy. In *IMA International Conference on Cryptography and Coding* (2001), Springer, pp. 222–238.

[64] Daemen, J., and Rijmen, V. AES and the wide trail design strategy. In *Advances in Cryptology - EUROCRYPT 2002* (2002), pp. 108–109.

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

[66] Daemen, J., and Rijmen, V. Plateau characteristics. *IET Information Security 1*, 1 (2007), 11–17.

[67] Daum, M. *Cryptanalysis of Hash functions of the MD4-family.* PhD thesis, Ruhr-Universität Bochum, 2005.

[68] De Canniere, C. Trivium: A stream cipher construction inspired by block cipher design principles. In *International Conference on Information Security* (2006), Springer, pp. 171–186.

[69] De Cannière, C. *Analysis and design of symmetric encryption algorithms.* PhD thesis, KU Leuven, 2007.

[70] De Moura, L., and Bjørner, N. Z3: An efficient SMT solver. *Tools and Algorithms for the Construction and Analysis of Systems* (2008), 337–340.

[71] De Witte, G., Ashur, T., and Liu, Y. An automated tool for Rotational-XOR cryptanalysis of ARX-based primitives. In *38th Symposium on Information Theory in the Benelux* (2017).

[72] Derbez, P., Fouque, P., and Jean, J. Improved key recovery attacks on reduced-round AES in the single-key setting. In *Advances in Cryptology - EUROCRYPT 2013* (2013), pp. 371–387.

[73] Dinu, D., Perrin, L., Udovenko, A., Velichkov, V., Grossschädl, J., and Biryukov, A. Design strategies for ARX with provable bounds: SPARX and LAX. In *Advances in Cryptology - ASIACRYPT 2016* (2016), Springer, pp. 484–513.

[74] Dinur, I. Improved differential cryptanalysis of round-reduced SPECK. In *International Workshop on Selected Areas in Cryptography - SAC 2014*. Springer, 2014, pp. 147–164.

[75] Dinur, I., Liu, Y., Meier, W., and Wang, Q. Optimized interpolation attacks on LowMC. In *Advance in Cryptology - ASIACRYPT 2015* (2015), Springer, pp. 535–560.

[76] Dinur, I., and Shamir, A. Cube attacks on tweakable black box polynomials. In *Advances in Cryptology - EUROCRYPT 2009* (2009), Springer-Verlag, pp. 278–299.

[77] Dobbertin, H., Bosselaers, A., and Preneel, B. RIPEMD-160: A strengthened version of RIPEMD. In *International Workshop on Fast Software Encryption - FSE'96* (1996), Springer, pp. 71–82.

[78] Dubuc, S. Characterization of linear structures. *Designs, Codes and Cryptography 22*, 1 (2001), 33–45.

[79] Dunkelman, O., Indesteege, S., and Keller, N. A differential-linear attack on 12-round Serpent. In *Progress in Cryptology - INDOCRYPT 2008*. Springer, 2008, pp. 308–321.

[80] Durak, F. B., and Vaudenay, S. Breaking the FF3 format-preserving encryption standard over small domains. In *Advances in Cryptology - CRYPTO 2017* (2017), Springer, pp. 679–707.

[81] Durumeric, Z., Kasten, J., Adrian, D., Halderman, J. A., Bailey, M., Li, F., Weaver, N., Amann, J., Beekman, J., Payer, M., and Paxson, V. The matter of Heartbleed. In *Proceedings of the 2014 Conference on Internet Measurement Conference* (2014), ACM, pp. 475–488.

[82] Dworkin, M. Recommendation for block cipher modes of operation: methods for format-preserving encryption. *NIST Special Publication 800* (2013), 38G.

[83] ETSI/SAGE. Specification of the 3GPP confidentiality and integrity algorithms UEA2 & UIA2. document 2: SNOW 3G specification. Tech. rep., 2006.

[84] Evertse, J.-H. Linear structures in blockciphers. In *Advances in Cryptology - EUROCRYPT'87* (1987), Springer, pp. 249–266.

[85] Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., and Walker, J. The Skein hash function family. *Submission to NIST* (2010).

[86] FIPS. 180-2: Secure hash standard (SHS). *US Department of Commerce, National Institute of Standards and Technology (NIST)* (2012).

[87] Fu, K., Wang, M., Guo, Y., Sun, S., and Hu, L. MILP-based automatic search algorithms for differential and linear trails for SPECK. In *International Workshop on Fast Software Encryption - FSE 2016* (2016), Springer, pp. 268–288.

[88] Fuhr, T. Finding second preimages of short messages for Hamsi-256. In *Advances in Cryptology - ASIACRYPT 2010* (2010), Springer, pp. 20–37.

[89] Ganesh, V., and Dill, D. L. A decision procedure for bit-vectors and arrays. In *International Conference on Computer Aided Verification - CAV 2007* (2007), pp. 519–531.

[90] Gauravaram, P., Knudsen, L. R., Matusiewicz, K., Mendel, F., Rechberger, C., Schläffer, M., and Thomsen, S. S. Grøstl-a SHA-3 candidate. In *Dagstuhl Seminar Proceedings* (2009), Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[91] Gérard, B., Grosso, V., Naya-Plasencia, M., and Standaert, F.-X. Block ciphers that are easier to mask: How far can we go? In *International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2013* (2013), Springer, pp. 383–399.

[92] Gerault, D., Minier, M., and Solnon, C. Constraint programming models for chosen key differential cryptanalysis. In *International Conference on Principles and Practice of Constraint Programming* (2016), Springer, pp. 584–601.

[93] Gong, Z., Nikova, S., and Law, Y. W. KLEIN: A new family of lightweight block ciphers. In *RFIDSec 2011* (2011), Springer, pp. 1–18.

[94] Grassi, L., Rechberger, C., and Rønjom, S. Subspace trail cryptanalysis and its applications to AES. *IACR Transactions on Symmetric Cryptology 2016*, 2 (2017), 192–225.

[95] Grassi, L., Rechberger, C., Rotaru, D., Scholl, P., and Smart, N. P. MPC-friendly symmetric key primitives. In *ACM SIGSAC Conference on Computer and Communications Security - CCS 2016* (2016), ACM, pp. 430–443.

[96] Grosso, V., Leurent, G., Standaert, F.-X., and Varici, K. LS-designs: Bitslice encryption for efficient masked software implementations. In *International Workshop on Fast Software Encryption - FSE 2014* (2014), Springer, pp. 18–37.

[97] Grosso, V., Leurent, G., Standaert, F.-X., Varici, K., Durvaux, F., Gaspar, L., and Kerckhof, S. SCREAM & iSCREAM side-channel resistant authenticated encryption with masking. *Submission to CAESAR* (2014).

[98] Guo, J., Jean, J., Nikolic, I., Qiao, K., Sasaki, Y., and Sim, S. M. Invariant subspace attack against Midori64 and the resistance criteria for S-box designs. *IACR Transactions on Symmetric Cryptology 2016*, 1 (2016), 33–56.

[99] Guo, J., Jean, J., Nikolić, I., and Sasaki, Y. Meet-in-the-middle attacks on generic Feistel constructions. In *Advances in Cryptology – ASIACRYPT 2014* (2014), Springer, pp. 458–477.

[100] Guo, J., Liu, M., and Song, L. Linear structures: Applications to cryptanalysis of round-reduced Keccak. In *Advances in Cryptology - ASIACRYPT 2016* (2016), Springer, pp. 249–274.

[101] Guo, J., Peyrin, T., and Poschmann, A. The PHOTON family of lightweight hash functions. In *Advances in Cryptology - CRYPTO 2011* (2011), Springer, pp. 222–239.

[102] Guo, J., Peyrin, T., Poschmann, A., and Robshaw, M. The LED block cipher. In *International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2011* (2011), vol. 6917, Springer, p. 326.

[103] Hall, C., Kelsey, J., Rijmen, V., Schneier, B., and Wagner, D. Cryptanalysis of SPEED. In *International Workshop on Selected Areas in Cryptography - SAC'98* (1998), pp. 319–338.

[104] Hamann, M., Krause, M., and Meier, W. LIZARD - a lightweight stream cipher for power-constrained devices. *IACR Transactions on Symmetric Cryptology 2017*, 1 (2017), 45–79.

[105] Hell, M., Johansson, T., Maximov, A., and Meier, W. The Grain family of stream ciphers. In *New Stream Cipher Designs*. Springer, 2008, pp. 179–190.

[106] Hermelin, M., Cho, J., and Nyberg, K. Multidimensional linear cryptanalysis of reduced round Serpent. In *Information Security and Privacy* (2008), Springer, pp. 203–215.

[107] Hermelin, M., and Nyberg, K. Linear cryptanalysis using multiple linear approximations. `https://eprint.iacr.org/2011/093`, 2011.

[108] Hoang, V. T., Morris, B., and Rogaway, P. An enciphering scheme based on a card shuffle. In *Advances in Cryptology - CRYPTO 2012*. Springer, 2012, pp. 1–13.

[109] Hoang, V. T., and Rogaway, P. On generalized Feistel networks. In *Advances in Cryptology – CRYPTO 2010* (2010), Springer, pp. 613–630.

[110] Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B.-S., Lee, C., Chang, D., Lee, J., Jeong, K., et al. HIGHT: A new block cipher suitable for low-resource device. In *International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2006* (2006), Springer, pp. 46–59.

[111] Jakobsen, T., and Knudsen, L. R. The interpolation attack on block ciphers. In *International Workshop on Fast Software Encryption - FSE'97* (1997), Springer, p. 28.

[112] Joux, A. *Algorithmic cryptanalysis*. CRC Press, 2009.

[113] Kaplan, M., Leurent, G., Leverrier, A., and Naya-Plasencia, M. Breaking symmetric cryptosystems using quantum period finding. In *Advances in Cryptology - CRYPTO 2016* (2016), Springer, pp. 207–237.

[114] Kaplan, M., Leurent, G., Leverrier, A., and Naya-Plasencia, M. Quantum differential and linear cryptanalysis. *IACR Transactions on Symmetric Cryptology 2016*, 1 (2016), 71–94.

[115] Karpman, P., Peyrin, T., and Stevens, M. Practical free-start collision attacks on 76-step SHA-1. In *Advances in Cryptology - CRYPTO 2015* (2015), pp. 623–642.

[116] Kerdock, A. A class of low-rate non-linear binary codes. *Information and Control 20* (1972), 182–187.

[117] Khovratovich, D., and Nikolić, I. Rotational cryptanalysis of ARX. In *International Workshop on Fast Software Encryption - FSE 2010* (2010), Springer, pp. 333–346.

[118] Khovratovich, D., Nikolić, I., Pieprzyk, J., Sokołowski, P., and Steinfeld, R. Rotational cryptanalysis of ARX revisited. In *International Workshop on Fast Software Encryption - FSE 2015* (2015), Springer, pp. 519–536.

[119] Khovratovich, D., Nikolić, I., and Rechberger, C. Rotational rebound attacks on reduced Skein. In *Advances in Cryptology - ASIACRYPT 2010* (2010), Springer, pp. 1–19.

[120] Klimov, A., and Shamir, A. A new class of invertible mappings. In *International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2002* (2002), pp. 470–483.

[121] Knudsen, L. DEAL - a 128-bit block cipher. *Complexity 258*, 2 (1998), 216.

[122] Knudsen, L., and Wagner, D. Integral cryptanalysis. In *International Workshop on Fast Software Encryption - FSE 2002* (2002), Springer, pp. 629–632.

[123] Knudsen, L. R. Iterative characteristics of DES and $s^2$-DES. In *Advances in Cryptology - CRYPTO'92* (1992), pp. 497–511.

[124] Knudsen, L. R. Truncated and higher-order differentials. In *International Workshop on Fast Software Encryption - FSE'94* (1994), Springer, pp. 196–211.

[125] Knudsen, L. R., Leander, G., Poschmann, A., and Robshaw, M. J. PRINTcipher: A block cipher for IC-printing. In *International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2010* (2010), Springer, pp. 16–32.

[126] Knudsen, L. R., and Rijmen, V. Known-key distinguishers for some block ciphers. In *Advances in Cryptology - ASIACRYPT 2007* (2007), Springer, pp. 315–324.

[127] Knudsen, L. R., Rijmen, V., Rivest, R. L., and Robshaw, M. J. On the design and security of RC2. In *International Workshop on Fast Software Encryption - FSE'98* (1998), Springer, pp. 206–221.

[128] Kölbl, S. CryptoSMT: An easy to use tool for cryptanalysis of symmetric primitives. `https://github.com/kste/cryptosmt`, 2015.

[129] Kölbl, S., Lauridsen, M. M., Mendel, F., and Rechberger, C. Haraka v2 - efficient short-input hashing for post-quantum applications. *IACR Transactions on Symmetric Cryptology 2016*, 2 (2017), 1–29.

[130] Kölbl, S., Leander, G., and Tiessen, T. Observations on the SIMON block cipher family. In *Advance in Cryptology - CRYPTO 2015* (2015), Springer, pp. 161–185.

[131] Lai, X. Higher order derivatives and differential cryptanalysis. In *Communications and Cryptography*. Springer, 1994, pp. 227–233.

[132] Lai, X., Massey, J., and Murphy, S. Markov ciphers and differential cryptanalysis. In *Advances in Cryptology - EUROCRYPT'91* (1991), Springer, pp. 17–38.

[133] Lai, X., and Massey, J. L. A proposal for a new block encryption standard. In *Advances in Cryptology - EUROCRYPT '90* (Berlin, Heidelberg, 1991), I. B. Damgård, Ed., Springer Berlin Heidelberg, pp. 389–404.

[134] Lallemand, V., and Naya-Plasencia, M. Cryptanalysis of KLEIN. In *International Workshop on Fast Software Encryption - FSE 2014* (2014), pp. 451–470.

[135] Langford, S., and Hellman, M. Differential-linear cryptanalysis. In *Advances in Cryptology - CRYPTO'94* (1994), Springer, pp. 17–25.

[136] Leander, G., Abdelraheem, M., AlKhzaimi, H., and Zenner, E. A cryptanalysis of PRINTcipher: the invariant subspace attack. In *Advances in Cryptology - CRYPTO 2011* (2011), Springer, pp. 206–221.

[137] Leander, G., Minaud, B., and Rønjom, S. A generic approach to invariant subspace attacks: Cryptanalysis of Robin, iSCREAM and Zorro. In *Advances in Cryptology - EUROCRYPT 2015* (2015), Springer, pp. 254–283.

[138] Leander, G., and Poschmann, A. On the classification of 4 bit Sboxes. In *International Workshop on the Arithmetic of Finite Fields* (2007), Springer, pp. 159–176.

[139] Leurent, G. Analysis of differential attacks in ARX constructions. In *Advances in Cryptology - ASIACRYPT 2012* (2012), pp. 226–243.

[140] Leurent, G. ARXtools: a toolkit for ARX analysis. `https://who.rocq.inria.fr/Gaetan.Leurent/arxtools.html`, 2012.

[141] Leurent, G. Construction of differential characteristics in ARX designs application to Skein. In *Advances in Cryptology - CRYPTO 2013* (2013), Springer, pp. 241–258.

[142] Li, C., and Wang, Q. Design of lightweight linear diffusion layers from near-MDS matrices. *IACR Transactions on Symmetric Cryptology 2017*, 1 (2017), 129–155.

[143] Lint, J. V. Kerdock codes and Preparata codes. *Congressus Numerantium 39* (1983), 25–51.

[144] Lipmaa, H., and Moriai, S. Efficient algorithms for computing differential properties of addition. In *International Workshop on Fast Software Encryption - FSE 2001* (2001), Springer, pp. 336–350.

[145] Lipmaa, H., Wallén, J., and Dumas, P. On the additive differential probability of exclusive-or. In *International Workshop on Fast Software Encryption - FSE 2004* (2004), Springer, pp. 317–331.

[146] Liu, Y., De Witte, G., Ranea, A., and Ashur, T. Rotational-XOR cryptanalysis of reduced-round SPECK. *IACR Transactions on Symmetric Cryptology 2017*, 3 (2017), 24–36.

[147] Liu, Y., Qu, L., and Li, C. New constructions of systematic authentication codes from three classes of cyclic codes. *Advances in Mathematics of Communications 12(1)* (2018), 1–16.

[148] Liu, Y., and Rijmen, V. New observations on invariant subspace attack. *Information Processing Letters 138* (2018), 27 – 30.

[149] Liu, Y., Rijmen, V., and Leander, G. Nonlinear diffusion layers. *To appear in Design, Codes and Cryptography* (2018).

[150] Liu, Y., Wang, Q., and Rijmen, V. Automatic search of linear trails in ARX with applications to SPECK and Chaskey. In *International Conference on Applied Cryptography and Network Security - ACNS 2016* (2016), Springer, pp. 485–499.

[151] Luby, M., and Rackoff, C. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing 17*, 2 (1988), 373–386.

[152] Matsui, M. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology - EUROCRYPT'93* (1994), Springer, pp. 386–397.

[153] Matsui, M. New block encryption algorithm MISTY. In *International Workshop on Fast Software Encryption - FSE'97* (1997), vol. 1267, Springer, pp. 54–68.

[154] Matsui, M., and Tokita, T. MISTY, KASUMI and Camellia cipher algorithm development. *Mitsubishi Electric Advance (Mitsubishi Electric corp.) 100* (2001), 2–8.

[155] Méaux, P., Journault, A., Standaert, F.-X., and Carlet, C. Towards stream ciphers for efficient FHE with low-noise ciphertexts. In *Advances in Cryptology - EUROCRYPT 2016* (2016), Springer, pp. 311–343.

[156] Mendel, F., Rechberger, C., Schläffer, M., and Thomsen, S. S. The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In *International Workshop on Fast Software Encryption - FSE 2009* (2009), pp. 260–276.

[157] Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M. J., Terrel, A. R., Roučka, v., Saboo, A., Fernando, I., Kulal, S., Cimrman, R., and Scopatz, A. SymPy: symbolic computing in Python. *PeerJ Computer Science 3* (2017).

[158] Morris, B., Rogaway, P., and Stegers, T. How to encipher messages on a small domain. In *Advances in Cryptology - CRYPTO 2009*. Springer, 2009, pp. 286–302.

[159] Mouha, N., Mennink, B., Van Herrewege, A., Watanabe, D., Preneel, B., and Verbauwhede, I. Chaskey: An efficient MAC algorithm for 32-bit microcontrollers. In *International Workshop on Selected Areas in Cryptography - SAC 2014*. Springer, 2014, pp. 306–323.

[160] Mouha, N., Velichkov, V., De Canniere, C., and Preneel, B. The differential analysis of S-functions. In *International Workshop on Selected Areas in Cryptography* (2010), Springer, pp. 36–56.

[161] Mouha, N., Wang, Q., Gu, D., and Preneel, B. Differential and linear cryptanalysis using mixed-integer linear programming. In *Information Security and Cryptology - Inscrypt 2011* (2012), Springer, pp. 57–76.

[162] Nakahara Jr, J. 3D: A three-dimensional block cipher. In *International Conference on Cryptology and Network Security* (2008), Springer, pp. 252–267.

[163] NBS. Data encryption standard. *Federal Information Processing Standards Publication 46, US Department of Commerce* (1977).

[164] Needham, R. M., and Wheeler, D. J. TEA extensions. Tech. rep., 1997.

[165] Nikova, S., Rechberger, C., and Rijmen, V. Threshold implementations against side-channel attacks and glitches. In *International Conference on Information and Communications Security* (2006), Springer, pp. 529–545.

[166] Nordstrom, A., and Robinson, J. An optimum nonlinear code. *Information and Control 11* (1967), 613–616.

[167] Nyberg, K. Linear approximation of block ciphers. In *Advances in Cryptology - EUROCRYPT'94* (1995), Springer, pp. 439–444.

[168] Nyberg, K., and Wallén, J. Improved linear distinguishers for SNOW 2.0. In *International Workshop on Fast Software Encryption - FSE 2006* (2006), Springer, pp. 144–162.

[169] Paul, S., and Preneel, B. Solving systems of differential equations of addition. In *Australasian Conference on Information Security and Privacy* (2005), Springer, pp. 75–88.

[170] Prud'homme, C., Fages, J.-G., and Lorca, X. *Choco Documentation.* TASC, INRIA Rennes, 2016.

[171] Ranea, A., Liu, Y., and Ashur, T. An easy-to-use tool for rotational-XOR cryptanalysis of ARX block ciphers. *Proceedings of the Romanian Academy, Series A 18.3 (2017): 307-316*.

[172] Rijmen, V. *Cryptanalysis and design of iterated block ciphers.* PhD thesis, KU Leuven, 1997.

[173] Ristenpart, T., and Yilek, S. The mix-and-cut shuffle: small-domain encryption secure against N queries. In *Advances in Cryptology - CRYPTO 2013*. Springer, 2013, pp. 392–409.

[174] Sasaki, Y., and Todo, Y. New impossible differential search tool from design and cryptanalysis aspects. In *Advances in Cryptology - EUROCRYPT 2017* (2017), Springer, pp. 185–215.

[175] Schulte-Geers, E. On CCZ-equivalence of addition mod $2^n$. *Designs, Codes and Cryptography 66*, 1-3 (2013), 111–127.

[176] Shamir, A. How to share a secret. *Communications of the ACM 22*, 11 (1979), 612–613.

[177] Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., and Shirai, T. Piccolo: An ultra-lightweight blockcipher. In *International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2011* (2011), vol. 6917, Springer, pp. 342–357.

[178] Shimoyama, T., Moriai, S., and Kaneko, T. Improving the higher-order differential attack and cryptanalysis of the KN cipher. In *International Workshop on Information Security* (1997), Springer, pp. 32–42.

[179] Shirai, T., Shibutani, K., Akishita, T., Moriai, S., and Iwata, T. The 128-bit blockcipher CLEFIA. In *International Workshop on Fast Software Encryption - FSE 2007* (2007), vol. 4593, Springer, pp. 181–195.

[180] Sinz, C. Towards an optimal CNF encoding of boolean cardinality constraints. In *Principles and Practice of Constraint Programming - CP 2005*. Springer, 2005, pp. 827–831.

[181] Soos, M. A blog about SAT solving and cryptography. `http://www.msoos.org`.

[182] Soos, M., Nohl, K., and Castelluccia, C. Extending SAT solvers to cryptographic problems. In *Theory and Applications of Satisfiability Testing - SAT 2009*. Springer, 2009, pp. 244–257.

[183] Standaert, F.-X., Piret, G., Gershenfeld, N., and Quisquater, J.-J. SEA: A scalable encryption algorithm for small embedded applications. In *Smart Card Research and Advanced Applications*. Springer, 2006, pp. 222–236.

[184] Stevens, M., Bursztein, E., Karpman, P., Albertini, A., and Markov, Y. The first collision for full SHA-1. In *Advances in Cryptology - CRYPTO 2017* (2017), pp. 570–596.

[185] Strassen, V. Gaussian elimination is not optimal. *Numerische mathematik 13*, 4 (1969), 354–356.

[186] Sun, B., Liu, M., Guo, J., Rijmen, V., and Li, R. Provable security evaluation of structures against impossible differential and zero-correlation linear cryptanalysis. In *Advances in Cryptology - EUROCRYPT 2016* (2016), pp. 196–213.

[187] Sun, B., Liu, Z., Rijmen, V., Li, R., Cheng, L., Wang, Q., AlKhzaimi, H., and Li, C. Links among impossible differential, integral and zero-correlation linear cryptanalysis. In *Advances in Cryptology - CRYPTO 2015* (2015), pp. 95–115.

[188] Sun, S., Gerault, D., Lafourcade, P., Yang, Q., Todo, Y., Qiao, K., and Hu, L. Analysis of AES, Skinny, and others with constraint programming. *IACR Transactions on Symmetric Cryptology 2017*, 1 (2017), 281–306.

[189] Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., and Song, L. Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES (L) and other bit-oriented block ciphers. In *Advances in Cryptology - ASIACRYPT 2014*. Springer, 2014, pp. 158–178.

[190] Suzaki, T., Minematsu, K., Morioka, S., and Kobayashi, E. TWINE: A lightweight, versatile block cipher. In *ECRYPT Workshop on Lightweight Cryptography* (2011), vol. 2011.

[191] Tiessen, T. Polytopic cryptanalysis. In *Advances in Cryptology – EUROCRYPT 2016* (2016), Springer, pp. 214–239.

[192] Todo, Y. Integral cryptanalysis on full MISTY1. *Journal of Cryptology 30*, 3 (2017), 920–959.

[193] Todo, Y., Leander, G., and Sasaki, Y. Nonlinear invariant attack. In *Advances in Cryptology – ASIACRYPT 2016* (2016), Springer, pp. 3–33.

[194] Tolba, M., Abdelkhalek, A., and Youssef, A. M. Truncated and multiple differential cryptanalysis of reduced round Midori128. In *International Conference on Information Security - ISC 2016* (2016), pp. 3–17.

[195] Van Lint, J. H. *Introduction to coding theory*, vol. 86. Springer Science & Business Media, 2012.

[196] Vanhoef, M., and Piessens, F. Key reinstallation attacks: Forcing nonce reuse in WPA2. In *ACM SIGSAC Conference on Computer and Communications Security - CCS 2017* (2017), ACM, pp. 1313–1328.

[197] Velichkov, V. YAARX: yet another toolkit for the analysis of ARX cryptographic algorithms. `https://github.com/vesselinux/yaarx`, 2016.

[198] Wagner, D. The boomerang attack. In *International Workshop on Fast Software Encryption - FSE '99* (1999), Springer-Verlag, pp. 156–170.

[199] Wallén, J. Linear approximations of addition modulo $2^n$. In *International Workshop on Fast Software Encryption - FSE 2011* (2003), Springer, pp. 261–273.

[200] Wang, G., Keller, N., and Dunkelman, O. The delicate issues of addition with respect to XOR differences. In *International Workshop on Selected Areas in Cryptography - SAC 2007* (2007), pp. 212–231.

[201] Wang, M., Sun, Y., Tischhauser, E., and Preneel, B. A model for structure attacks, with applications to PRESENT and Serpent. In *International Workshop on Fast Software Encryption - FSE 2012* (2012), pp. 49–68.

[202] Wang, X., Yin, Y. L., and Yu, H. Finding collisions in the full SHA-1. In *Advances in Cryptology - CRYPTO 2005* (2005), pp. 17–36.

[203] Wang, X., and Yu, H. How to break MD5 and other hash functions. In *Advances in Cryptology - EUROCRYPT 2005* (2005), pp. 19–35.

[204] West, D. B. *Introduction to graph theory*, vol. 2. Prentice hall Upper Saddle River, 2001.

[205] Wheeler, D. J., and Needham, R. M. TEA, a tiny encryption algorithm. In *International Workshop on Fast Software Encryption - FSE'95* (1995), Springer, pp. 363–366.

[206] Wolfram, S. *Theory and applications of cellular automata*, vol. 1. World scientific Singapore, 1986.

[207] Wu, H. ACORN: a lightweight authenticated cipher (v3). *Submission to CAESAR* (2016).

[208] Wu, W., and Zhang, L. LBlock: a lightweight block cipher. In *International Conference on Applied Cryptography and Network Security - ACNS 2011* (2011), Springer, pp. 327–344.

[209] Xiang, Z., Zhang, W., Bao, Z., and Lin, D. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In *Advances in Cryptology - ASIACRYPT 2016* (2016), Springer, pp. 648–678.

[210] Yao, A. C.-C. How to generate and exchange secrets. In *Annual Symposium on Foundations of Computer Science* (1986), IEEE, pp. 162–167.

[211] Yao, Y., Zhang, B., and Wu, W. Automatic search for linear trails of the SPECK family. In *International Information Security Conference - ISC 2015* (2015), Springer, pp. 158–176.

[212] Zhang, W., Bao, Z., Lin, D., Rijmen, V., Yang, B., and Verbauwhede, I. RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. *SCIENCE CHINA Information Sciences 58*, 12 (2015), 1–15.

# Curriculum Vitae

Yunwen Liu was born on January 26, 1991 in Shiyan, China. She studied Applied Mathematics in National University of Defence Technology, Changsha, China. In 2012, she obtained her Bachelor degree and a postgraduate recommendation without examination (equivalent to the highest distinction, ranking top 2 in the department). In 2014, she received a Master degree in Applied Mathematics with her master thesis focused on systematic authentication codes constructed from cyclic codes.

Starting from September 29, 2014, Yunwen is a doctoral student in the research group COSIC (Computer Security and Industrial Cryptography) at ESAT (Department of Electrical Engineering) of KU Leuven, Belgium. Her research is supported by a doctoral scholarship from CSC (China Scholarship Council). During her doctoral study, Yunwen has been active in research and academic activities, and coauthored more than twelve research papers. She presented research results in FSE 2018, FSE 2017 and ACNS 2016, gave invited talks at ASK 2017 and the workshop on ZUC-256 and 5G in 2018.

# List of Publications

Journal Publications:

1. Yunwen Liu, Vincent Rijmen, and Gregor Leander. Nonlinear diffusion layers. To appear in Designs, Codes and Cryptography, 2018.

2. Yunwen Liu and Vincent Rijmen. New observations on invariant subspace attack. Information Processing Letters 138 (2018), 27-30.

3. Yunwen Liu, Longjiang Qu, Chao Li. New constructions of systematic authentication codes from three classes of cyclic codes. Advances in Mathematics of Communications 12 (1), 1-16, 2018.

4. Wei Li, Vincent Rijmen, Zhi Tao, Qingju Wang, Hua Chen, Yunwen Liu, Chaoyun Li, Ya Liu. Impossible meet-in-the-middle fault analysis on the LED lightweight cipher in VANETs. Science China Information Sciences 61 (3), 032110, 2018.

5. Yunwen Liu, Glenn De Witte, Adrián Ranea, and Tomer Ashur. Rotational-XOR cryptanalysis of reduced-round SPECK. IACR Transactions on Symmetric Cryptology 2017, 3 (2017), 24-36.

6. Adrián Ranea, Yunwen Liu, and Tomer Ashur. An easy-to-use tool for rotational-XOR cryptanalysis of ARX block ciphers. Proceedings of the Romanian Academy, Series A 18.3 (2017), 307-316.

7. Tomer Ashur and Yunwen Liu. Rotational cryptanalysis in the presence of constants. IACR Transactions on Symmetric Cryptology 2016, 1 (2016), 57-70.

International Conferences:

1. Yunwen Liu, Yu Sasaki, Ling Song, Gaoli Wang. Cryptanalysis of reduced sLiSCP permutation in sponge-hash and duplex-AE modes. In Selected Areas in Cryptography - SAC 2018, Springer.

2. Yunwen Liu, Qingju Wang, and Vincent Rijmen. Automatic search of linear trails in ARX with applications to Speck and Chaskey. In International Conference on Applied Cryptography and Network Security - ACNS 2016, Springer, 485-499.

3. Hua Chen, Jingyi Feng, Vincent Rijmen, Yunwen Liu, Limin Fan, Wei Li. Improved fault analysis on Simon block cipher family. In Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2016, 16-24.

4. Itai Dinur, Yunwen Liu, Willi Meier, and Qingju Wang. Optimized interpolation attacks on LowMC. In Advance in Cryptology - ASIACRYPT 2015, Springer, 535-560.

National Conferences:

1. Glenn De Witte, Tomer Ashur, and Yunwen Liu. An automated tool for rotational-XOR cryptanalysis of ARX-based primitives. In 38th Symposium on Information Theory in the Benelux (2017).

Manuscripts:

1. Bing Sun, Yunwen Liu, Guoqiang Liu, Chao Li and Shaojing Fu. The phantom of differential characteristics.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
ESAT - COSIC
Kasteelpark Arenberg 10 - bus 2452
B-3001 Leuven
yunwen.liu@esat.kuleuven.be
https://www.esat.kuleuven.be/cosic/