Cryptanalysis of Some Block Cipher Constructions

Ahmed Abdelkhalek

A Thesis

in

The Concordia Institute

for

Information Systems Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy (Information Systems Engineering) at
Concordia University
Montreal, Québec, Canada

October 2017

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Ahmed Abdelkhalek

Entitled: Cryptanalysis of Some Block Cipher Constructions

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Information Systems Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Rabin Raut

_____ External Examiner
Dr. Arash Reyhani-Masouleh

_____ External to Program
Dr. Walaa Hamouda

_____ Examiner
Dr. Chadi Assi

_____ Examiner
Dr. Lingyu Wang

_____ Thesis Supervisor
Dr. Amr M. Youssef

Approved by

_____
Dr. Chadi Assi, Graduate Program Director

December 8^{th}, 2017

_____
Dr. Amir Asif, Dean, Faculty of Engineering and Computer Science

Abstract

Cryptanalysis of Some Block Cipher Constructions

Ahmed Abdelkhalek, Ph.D.

Concordia University, 2017

When the public-key cryptography was introduced in the 1970s, symmetric-key cryptography was believed to soon become outdated. Nevertheless, we still heavily rely on symmetric-key primitives as they give high-speed performance. They are used to secure mobile communication, e-commerce transactions, communication through virtual private networks and sending electronic tax returns, among many other everyday activities. However, the security of symmetric-key primitives does not depend on a well-known hard mathematical problem such as the factoring problem, which is the basis of the RSA public-key cryptosystem. Instead, the security of symmetric-key primitives is evaluated against known cryptanalytic techniques. Accordingly, the topic of furthering the state-of-the-art of cryptanalysis of symmetric-key primitives is an ever-evolving topic. Therefore, this thesis is dedicated to the cryptanalysis of symmetric-key cryptographic primitives. Our focus is on block ciphers as well as hash functions that are built using block ciphers. Our contributions can be summarized as follows:

First, we tackle the limitation of the current Mixed Integer Linear Programming (MILP) approaches to represent the differential propagation through large S-boxes. Indeed, we present a novel approach that can efficiently model the Difference Distribution Table (DDT) of large S-boxes, i.e., 8-bit S-boxes. As a proof of the validity and efficiency of our approach, we apply

it on two out of the seven AES-round based constructions that were recently proposed in FSE 2016. Using our approach, we improve the lower bound on the number of active S-boxes of one construction and the upper bound on the best differential characteristic of the other.

Then, we propose meet-in-the-middle attacks using the idea of efficient differential enumeration against two Japanese block ciphers, i.e., Hierocrypt-L1 and Hierocrypt-3. Both block ciphers were submitted to the New European Schemes for Signatures, Integrity, and Encryption (NESSIE) project, selected as one of the Japanese e-Government recommended ciphers in 2003 and reselected in the candidate recommended ciphers list in 2013. We construct five S-box layer distinguishers that we use to recover the master keys of reduced 8 S-box layer versions of both block ciphers. In addition, we present another meet-in-the-middle attack on Hierocrypt-3 with slightly higher time and memory complexities but with much less data complexity.

Afterwards, we shift focus to another equally important cryptanalytic attack, i.e., impossible differential attack. SPARX-64/128 is selected among the SPARX family that was recently proposed to provide ARX based block cipher whose security against differential and linear cryptanalysis can be proven. We assess the security of SPARX-64/128 against impossible differential attack and show that it can reach the same number of rounds the division-based integral attack, proposed by the designers, can reach. Then, we pick Kiasu-BC as an example of a tweakable block cipher and prove that, on contrary to its designers' claim, the freedom in choosing the publicly known tweak decreases its security margin. Lastly, we study the impossible differential properties of the underlying block cipher of the Russian hash standard Streebog and point out the potential risk in using it as a MAC scheme in the secret-IV mode.

Acknowledgments

I would like to express my deepest appreciation to my supervisor Professor Amr Youssef. He enrolled me in his Crypto team despite my entirely different background. Throughout my PhD, he never failed to answer my constant stream of questions with patience and support. Without his guidance and endless motivation, this PhD would simply not have been.

I would also like to thank my lab mates for their friendship, support, and constructive feedback. Special thanks go to Mohamed Tolba for his help and fruitful personal and research discussions. His friendship made the PhD journey more enjoyable.

Last, but by no means least, I am deeply grateful to my wife and children, for their constant encouragement and unconditional love, and for enduring my late working hours, even on weekends at times. I would not be where I am in life without my mother and family members; thank you for believing in me and for always pushing me to excel in everything I do.

AHMED ABDELKHALEK

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Cryptology can be thought of as a coin where cryptography is one side and cryptanalysis is the other [139]. While cryptography deals with the design of algorithms achieving specific security goals, cryptanalysis studies attacks on such algorithms with the goal of violating their security claims. The set of security goals considered in modern cryptography varies according to its application and includes confidentiality, integrity, authenticity, anonymity and non-repudiation [112]. The building blocks of many current information systems that are used to provide such security goals are called cryptographic primitives.

Cryptographic primitives comprise symmetric-key or secret-key primitives, asymmetric-key or public-key primitives and unkeyed primitives (see Figure 1.1). In the symmetric-key primitives, a shared key that is kept secret within a restricted group is used. In the public-key primitives, each user has a pair of keys; one is kept private and never shared and the other is public. The unkeyed primitives, as their name suggests, do not use any secret key. After the invention of public-key cryptography in the mid 1970s, there was a misconception that symmetric-key primitives would become obsolete. However, they are still extensively used as they achieve high-speed performance or low-cost encryption, fast authentication, and efficient hashing. Nowadays,

symmetric-key primitives are used in securing many applications including cell phones communications, credit card transactions, and Wi-Fi connections and symmetric-key cryptology is an active research area as it has always been. In fact, there is still a compelling need to further the symmetric-key cryptology research. On the one hand, the huge demand for deploying resource-constrained, yet cryptographically-secure, devices such as RFID tags and wireless sensor nodes dictates the demand for low-cost primitives. On the other hand, the development of the state-of-the-art of cryptanalysis may threaten the security of some existing and widely used primitives. Therefore, the problem of studying new cryptographic techniques remains necessary to evaluate the existing cryptographic primitives and design new more efficient and secure ones.

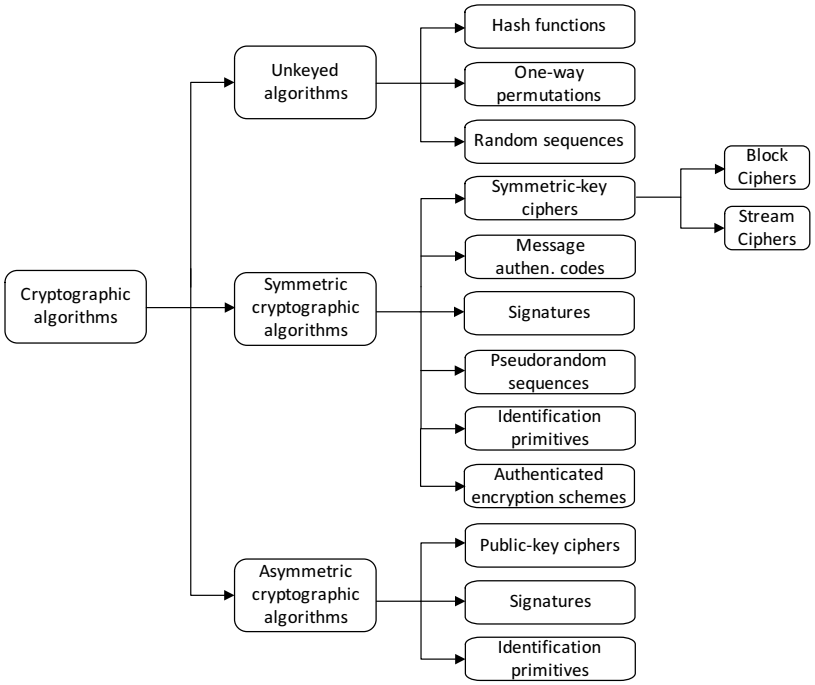

Figure 1.1: Cryptographic primitives classification [112]

1.2 Symmetric-key Cryptographic Primitives

A symmetric-key cryptographic primitive is a transformation that takes two inputs (data and a secret key) and maps them to one output. Three basic types that are of interest are: ciphers (either block or stream ciphers), Message Authentication Codes (MACs), and authenticated en-

cryption schemes. Strictly speaking, cryptographic hash functions are not symmetric-key cryptographic primitives, since they do not use a secret key. However, MACs can be constructed using hash functions. Moreover, hash functions can be built using block ciphers. Therefore, the properties of hash functions are closely related to those of symmetric-key cryptographic primitives and thus are also considered in our work. Symmetric-key cryptographic primitives can be employed to directly achieve data confidentiality, integrity and authenticity. Other security goals, e.g. non-repudiation, are achieved by cryptographic protocols that are based on symmetric-key cryptographic primitives.

Block ciphers and stream ciphers provide data confidentiality by encrypting a plaintext (the input message) to a ciphertext (the transformed output message). For a fixed key, block ciphers and stream ciphers become invertible mappings. On the one hand, a block cipher [89] can be thought of as a class of permutations on blocks of bits. When a particular key is selected, a specific permutation is chosen out of this class. The encryption process is done by applying this permutation to the plaintext yielding the ciphertext. The plaintext is restored via the decryption process by applying the inverse permutation to the ciphertext. On the other hand, a stream cipher [41] generates an infinite stream of pseudo-random digits (keystream) which is XORed with the plaintext (ciphertext) to produce the ciphertext (plaintext).

MACs [123] provide data integrity and authenticity. A MAC primitive takes the input message and the secret key and computes an authentication tag which is then sent along with the message. The recipient of the message then uses the same secret key to compute the authentication tag corresponding to the received message. If the calculated authentication tag matches the received one, then the sender is authenticated and the message integrity is verified. It is worth mentioning that, unlike block ciphers and stream ciphers, MAC primitives do not have to be invertible, as both users perform the same operation to compute or verify the authentication tag.

A hash function [122] is a primitive that transforms a message of any arbitrary, but usually limited, length into a hash value of fixed length. In other words, a hash function compresses a message and generates a digest that depends on all the bits of the message. Therefore, hash functions, similar to MAC primitives, can be used to provide data integrity. However, they do not use a secret key and thus anyone can generate the hash value for a given message. The cryptographic properties of hash functions vary according to the application that embodies them and can include preimage resistance, second preimage resistance and collision resistance.

Authenticated Encryption schemes (AE) [31] are symmetric-key cryptographic primitives that provide data confidentiality, integrity and authenticity at the same time. An extension of AE schemes is Authenticated Encryption with Associated Data (AEAD) schemes. AEAD schemes take an additional input, i.e., the Associated Data (AD) whose integrity and authenticity has to be ensured while being sent in clear without encryption. For example, routing information in headers of datagram packets that has to remain unencrypted for routing purposes and at the same time has to be authenticated to detect any tampering. The design and cryptanalysis of AE(AD) primitives have found renewed interest driven by the NIST-funded competition: Competition for Authenticated Encryption: Security, Applicability and Robustness (CAESAR) [39]. CAESAR was announced in 2013, and "*will identify a portfolio of authenticated ciphers that (1) offer advantages over AES-GCM and (2) are suitable for widespread adoption*" [39].

1.3 Cryptanalysis Approaches: A Brief Overview

Traditionally, the end-points of the communication channel were assumed trusted and thus the cryptographic primitives were supposed to be executed in a secure environment. In such conventional cryptographic model, known as the black-box attack model, the adversary can observe the output and/or input of the cryptographic primitive and can (adaptively) choose different types of input and/or output to the primitive. Additionally, the attacker can sometimes specify the relation between multiple keys utilized by the primitive. The goals of the attacker

vary according to the targeted cryptographic primitive, i.e., in the case of block and stream ciphers the goals include being able to distinguish the behavior of the specific cipher from the behavior of a random permutation, to encrypt/decrypt some messages without knowing the secret key or to recover information about the secret key. In MAC primitives, the attacker may additionally attempt to generate a valid authentication tag for a given or a chosen message without the knowledge of the secret key. In hash functions, the attacker may try to find two distinct messages that yield the same hash value (a collision) or a (second) preimage for a given message. In AE(AD) schemes, the goals of the attacker combine the goals achieved while attacking an encryption scheme and a MAC primitive.

Within the black-box attack model, one can distinguish between structural cryptanalysis [26], statistical cryptanalysis [149, 150] and algebraic cryptanalysis [44]. While structural cryptanalysis studies the properties of the high-level building blocks of the primitives, statistical cryptanalysis analyzes and exploits undesirable probabilistic properties of their low-level building blocks. Algebraic cryptanalysis uses algebraic representation of the primitives to build and solve linear and non-linear equations on input, output and key variables. In the next chapter, the different cryptanalysis techniques are discussed in more details.

1.4 Research Contributions and Outline

In this thesis, we analyze symmetric-key cryptographic primitives. In particular, we study the security of block ciphers and hash functions constructed using block ciphers.

- In chapter 2, block ciphers are formally introduced and an overview of the conventional cryptanalysis techniques including differential cryptanalysis, linear cryptanalysis and Meet-in-the-Middle (MitM) cryptanalysis is provided.

- In chapter 3, we propose a new Mixed Integer Linear Programming (MILP) model for the Difference Distribution Table (DDT) of large S-boxes, i.e., 8-bit S-boxes and use our

model to improve the bounds of two new constructions based on the AES-round function.

- In chapters 4 and 5, we propose key recovery meet-in-the-middle attacks on the Japanese block ciphers Hierocrypt-L1 and Hierocrypt-3.

- In chapter 6, we study the security of SPARX-64/128, which is a member of the SPARX family that followed the recently proposed long trail strategy. We assess its resistance against impossible differential cryptanalysis.

- In chapter 7, we analyze the security of one tweakable block cipher, Kiasu-BC against the impossible differential cryptanalysis.

- In chapter 8, we investigate the security of the underlying block cipher of the Russian standard hash function, Streebog, against impossible differential cryptanalysis.

Some of the above contributions have been published in [1–5]. Other works conducted during the tenure of this Ph.D. have been published in [6, 141–147].

Chapter 2

Background and Literature Review

In this chapter, we more formally introduce block ciphers. We also provide a high-level literature review on some of the relevant cryptanalytic techniques.

2.1 Block Ciphers Definition

A block cipher is a symmetric-key primitive with the purpose of protecting the secrecy of messages sent over an insecure channel. It is defined as follows:

Definition 2.1 (Block cipher [112, Definition 7.1]) *An n-bit block cipher is a function* $E : V_n \times \mathcal{K} \to V_n$*, such that for each key* $K \in \mathcal{K}$*, E(P,K) is an invertible mapping (the encryption function for K) from* V_n *to* V_n*, written* $E_K(P)$*. The inverse mapping is the decryption function, denoted* $D_K(C)$*. C =* $E_K(P)$ *denotes the ciphertext C results from encrypting plaintext P under K.*

A block cipher can be thought of as 2^k permutations on $\mathbb{F}_2^n$ out of its $2^n!$ distinct permutations. Hence, for a block cipher to be ideal, its 2^k permutations have to be chosen randomly out of all the $2^n!$ possible permutations.

Definition 2.2 (Ideal block cipher [30]) *An n-bit block cipher E is called ideal, if the family of*

2^k *permutations on* $\mathbb{F}_2^n$ *specified by E is selected uniformly at random from the set of all* $(2^n)!$ *distinct permutation on* $\mathbb{F}_2^n$.

An ideal block cipher as defined above is impractical to implement as it requires storing 2^k truly random selected permutations on $\mathbb{F}_2^n$ which is practically infeasible for typical secure values of n and k. Therefore, the design of block ciphers is inevitably highly structured as will be discussed in the next section.

2.2 Block Ciphers Design

In 1949, Shannon set the basis of the field of block ciphers design by putting forward the concepts of confusion and diffusion [129] that are still considered while designing new block ciphers nowadays. In simple terms, confusion means that each bit of the ciphertext should depend on as many bits as possible of the plaintext and the secret key. Confusion is usually achieved by using non-linear components in the block cipher such as substitution boxes (S-boxes) and/or modular addition. On the other hand, diffusion captures the influence of each plaintext bit and each key bit on the ciphertext bits. Ideally, flipping one bit of the plaintext/ciphertext should result in flipping each ciphertext/plaintext bit with probability 0.5. To achieve diffusion, a block cipher designer usually uses linear transformations or permutations working at bit or byte level.

Today, the most efficient block ciphers are built by iterating a simple key-dependent round function. Such a simple key-dependent round function acts individually as a weak block cipher but iterating it several times achieves a high degree of confusion and diffusion and thus may yield a strong block cipher. Figure 2.1 gives a schematic view of an iterative block cipher. More formally, an iterative block cipher is defined as follows:

Definition 2.3 (Iterative block cipher [33, Definition 4]) *An n-bit block cipher E is called an iterative block cipher with r rounds if for each key it can be represented as a composition of*

keyed round permutations, that is, if for each $K \in \mathbb{F}_2^k$*:*

$$E(K, .) = f_r(K_r, .) \circ f_{r-1}(K_{r-1}, .) \circ \ldots \circ f_2(K_2, .) \circ f_1(K_1, .),$$

where $\circ$ *denotes the superposition of permutations,* $f_i(K_i, .) : \mathbb{F}_2^n \to \mathbb{F}_2^n$ *are key dependent round permutations,* K_i *are round subkeys derived from the secret key K using a key schedule algorithm:*

$$ks : K \to (K_1, K_2, \ldots, K_{r-1}, K_r).$$

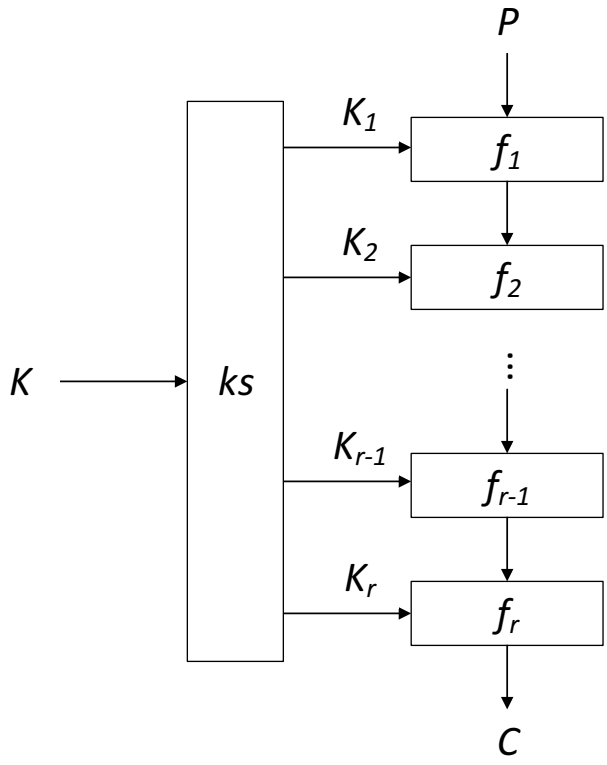

Figure 2.1: An iterative block cipher

If all the round permutations of an iterative block cipher are identical, i.e., $f_i = f$, then it is called an iterated block cipher. Typically, we still talk about an iterated block cipher even if the first and/or the last round are slightly different than the other rounds. Another distinguished

class of iterative block ciphers is the key-alternating block ciphers. In a such block cipher, the input I to the key-dependent permutation f_i is first XORed with the key K_i and then a key-independent permutation $f_i^{'}$ is applied, i.e., $f_i(K_i, I) = f_i^{'}(K_i \oplus I)$. A key-alternating block cipher that uses identical key-independent permutations is called a key-iterated block cipher.

There are so many constructions to build an iterative block cipher. Figure 2.2 shows the round functions of the two most widely accepted ones: Balanced Feistel Network (BFN) and Substitution-Permutation Network (SPN). In a BFN, the round function input is divided into two halves, i.e., L_i and R_i and the output L_{i+1} and R_{i+1} is computed as follows:

$$L_{i+1} = R_i,$$
$$R_{i+1} = L_i \oplus f(K_i, R_i),$$

where K_i is the i^{th} round subkey and f is the round function. A BFN has the advantage of reusing the implementation of the encryption function to implement the decryption one where the round subkeys are used in reverse. The former U.S. encryption standard DES [59], which is an iterated non-key-alternating block cipher, follows the BFN construction. The round function of an SPN block cipher consists of three layers (not necessarily in the following order): the parallel execution of S-boxes, linear permutation, and round subkey addition layer. The decryption function in SPN block ciphers is performed by inverting the encryption process while taking the round subkeys in reversed order. The advantage of SPN block ciphers lies in the fact that they tend to have good diffusion properties. The current U.S. encryption standard AES [115], which is a key-iterated block cipher, employs the SPN construction.

Other constructions to build iterative block ciphers include the Lai-Massey scheme followed by the IDEA block cipher [95], and Generalized Feistel Network (GFN) exemplified by the Twine block cipher [135].

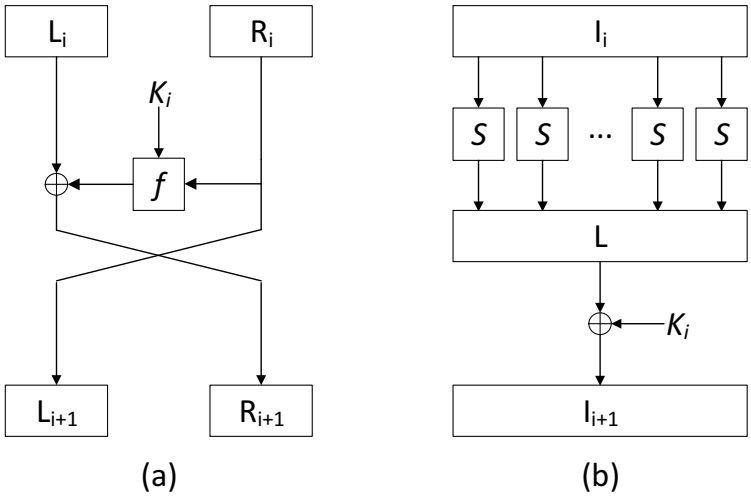

Figure 2.2: Round function of (a) Balanced Feistel Network, and (b) Substitution-Permutation Network

2.3 Block Ciphers Security

The notion of perfect security, the highest level of security, introduced by Shannon [129], is defined as follows:

Definition 2.4 (Perfect security [114, Definition 6]) *A cipher is called perfectly secure if the ciphertext does not reveal any information about the plaintext, or in other words, if the plaintext and the ciphertext are statistically independent.*

It was proved by Shannon that the entropy of the key in a perfectly secure cipher has to be at least as high as the entropy of the plaintext. This means that the length of the key in such ciphers must, at least, equal the length of the plaintext and it cannot be reused again. As large amounts of data need to be encrypted, perfectly secure block ciphers are impractical. Instead, practical block ciphers can achieve at most computational security defined as follows:

Definition 2.5 (Computational security [114, Definition 7]) *A block cipher E using a k-bit secret key is called computationally secure if there exist no attacks on E with a complexity less than the one of an exhaustive key search, i.e., 2^k, where the complexity of the attack comprises the time (work factor), memory (storage requirement), and data (type and amount of*

data) complexities required to perform the attack.

As the above definition implies, it is difficult to evaluate and/or prove that a block cipher is computationally (in)secure. Thus, the security of block ciphers is typically evaluated by considering their resistance against a large set of predefined cryptanalytic attacks and depends on two factors: the goal of the adversary and the assumed attack model, i.e., what the attacker is allowed to do to launch a certain attack.

2.3.1 Adversary's Goal

The goal of the adversary differs greatly with the application in which the cipher is deployed. Below, we give a hierarchical classification of these goals, following Knudsen [92], starting with the most powerful attack while assuming that the block cipher is used for encryption only:

- Total Break: The adversary recovers the secret key K.
- Global Deduction: The adversary can construct an algorithm that is functionally equivalent to E_K or D_K without knowing K.
- Local Deduction: The adversary can decrypt (encrypt) a previously unseen ciphertext (plaintext) without knowing K.
- Distinguishing Algorithm: The adversary can distinguish between the block cipher instantiated with a randomly chosen key and a randomly chosen permutation.

2.3.2 Attack Models

The attack models specify the capabilities of the adversary while trying to violate the security claims of a block cipher designer. In our work, we focus on the model assumed in conventional cryptography, i.e., the black-box attack model. In this model, the end-users of the communication channel are assumed to be trusted. This attack model can be classified to the following types

according to the operations on the input and/or output of the block cipher, which the adversary is allowed to perform:

- Ciphertext-only: The attacker can observe ciphertexts without any access to plaintexts. This is the weakest attack scenario and if a cipher is vulnerable to such attack, it is considered completely insecure.
- Known-plaintext: The attacker observes a number of plaintexts and their corresponding ciphertexts.
- Chosen-plaintext (ciphertext): The attacker can choose plaintexts (ciphertexts) to be encrypted (decrypted) before the attack and observes their corresponding ciphertexts (plaintexts) during the attack.
- Adaptively-chosen-plaintext (ciphertext): On top of the chosen-plaintext (ciphertext) capabilities mentioned above, the attacker can choose plaintexts (ciphertexts) during the attack based on some intermediate results obtained during the attack.
- Related-key: The attacker can encrypt (decrypt) plaintexts (ciphertexts) with the attacked key and other keys related to it, where such relation is known or even chosen by the attacker [19].

It is to be noted that the above classification is not comprehensive, for example, we can mix attacks from these classes yielding a, usually, more efficient one. In addition, in block cipher based hash functions, the key of the underlying block cipher is known or even under the control of the attacker and thus other attacks, such as the known key and the chosen key settings [91] can be relevant in this case.

2.4 Attacks on Block Ciphers

In this section, we discuss the most common attacks on block ciphers within the black-box attack model described above. The attacks in this model can be classified into generic and non-generic attacks. Generic attacks exploit the core parameters of the block cipher, e.g., the block size and key length. Non-generic attacks additionally exploit the block cipher internal structure and the specifications of its components.

Generic Attacks

Exhaustive key search (brute-force) attack: In this attack, all the possible values of the secret key are tested against a known plaintext/ciphertext pair. The time complexity of this attack to recover a k-bit secret key is about 2^k encryptions. If the attack results in more than one key, then it is repeated against additional plaintext/ciphertext pair. The unicity distance [112] determines the number of pairs needed to have a single key. Another variant of the brute-force attack is the so-called the dictionary attack which trades memory against time. In the dictionary attack, which is a chosen plaintext attack, the attacker first stores the encryption of a random plaintext under all the 2^k possible values of the key in a lookup table T. Then, in the actual attack, the chosen plaintext is encrypted using the secret key and the resulting ciphertext is looked up in T to identify the correct key.

Ciphertext-collision attack: First, the attacker gets $2^{n/2}$ random plaintext/ciphertext pairs encrypted using the secret key and stores them in a lookup table T, where n is the block size in bits. Afterwards, the attacker observes $2^{n/2}$ random ciphertexts and, due to the birthday paradox, one of these ciphertexts will match one of the ciphertexts stored in T with high probability. Then, the attacker instantly gets the corresponding plaintext as it is stored in T as well. In a variant of this attack, called the codebook attack, the attacker can store all 2^n plaintext/ciphertext pairs. Hence, any ciphertext can readily be decrypted by means of a single table lookup.

Time-memory trade-off attack [73, 119]: In this attack, the key space is broken into smaller sets and then each of these sets is exhaustively searched. In Hellman's time-memory trade-off attack, a k-bit key is found in $2^{2k/3}$ operations with $2^{2k/3}$ words of memory by using 2^k precomputations to create the offline lookup table.

Key-collision attack [20]: This attack resembles the ciphertext-collision attack mentioned above but here the attacker uses the birthday paradox to identify the correct key. First, the attacker encrypts a number of plaintexts, determined by the unicity distance, using $2^{k/2}$ random known keys and stores the keys and the corresponding ciphertexts in a lookup table T. Then, these plaintexts are encrypted using $2^{k/2}$ random secret keys. Due to the birthday paradox, one of these keys will match one of the known keys with a high probability resulting in the same ciphertext values.

Related-key cube-based attacks [40]: A variant of the cube attack [65] can be applied to all block ciphers in the related-key setting. To recover the secret key, such attack requires $\frac{2}{k}2^k$ encryptions, $\log k$ words of memory and negligible precomputations.

As the generic attacks are entirely independent on the block cipher design or its specification, the only countermeasure against these attacks is to increase the key and plaintext spaces. Currently, NIST's recommended key length specified in bits is a specific value from the set $\{80, 112, 128, 192, 256\}$ noting that an 80-bit key length is no longer considered sufficiently secure [117].

Non-generic Attacks

Non-generic attacks exploit particular properties of the block cipher design, i.e., its structure and internal components. Usually, in the non-generic attacks, the adversary first attempts to construct a distinguisher that covers as many rounds of the block cipher as possible and which holds with a relatively high probability. Then, this distinguisher is extended to a key-recovery

attack on a few additional rounds using partial decryption and/or encryption. First, the involved bits of the subkeys of the appended rounds are guessed. Then, the ciphertext and/or the plaintext is partially decrypted/encrypted using these guessed subkeys to calculate the intermediate state values at the ends of the distinguisher. If the subkeys are correctly guessed then the distinguisher should hold, otherwise, it fails.

Differential Cryptanalysis: Differential cryptanalysis is a chosen plaintext attack that was published in the academic literature by Biham and Shamir in 1990 by applying it on DES [25]. In differential cryptanalysis, the adversary looks at the input difference and at an intermediate difference after a few rounds. In other words, the distinguisher that differential cryptanalysis exploits is a differential [96], i.e., a pair of differences $(\Delta I_1, \Delta I_{s+1})$ that holds with a probability p significantly higher than 2^{-n}, where ΔI_1 is the input difference, and ΔI_{s+1} is the output difference after s rounds, and n is the block size in bits.

The data complexity, in terms of the number of chosen plaintext pairs, of a differential cryptanalysis is inversely proportional to the probability of the differential, i.e., the higher the probability of the differential is, the lower the data complexity of attack will be. However, in many ciphers, finding differentials that hold with high probability is not an easy task. Instead, one searches for differential trails where an s-round differential trail is a sequence of $s+1$ differences $(\Delta I_1, \Delta I_2, \ldots, \Delta I_s, \Delta I_{s+1})$, where ΔI_i is the input difference at round i. That is, while a differential cares about the input and output differences, a differential trail cares about how the difference is propagated throughout the block cipher components. Through non-linear components, the difference propagates with some probability, and it propagates deterministically through linear ones. Then, the probability of an s-round differential trail is approximated by $\prod_{i=1}^{s} p_i$, where p_i is the probability of this differential trail in round i. As an s-round differential $(\Delta I_1, \Delta I_{s+1})$ can be thought of as the set of all possible differential trails $(\Delta I_1, \ldots, \Delta I_{s+1})$ that have the same input and output differences, the probability of a differential is the sum of the probabilities of all its constituent differential trails. In general, the probability of a differential

can depend on the values of the key, the input and the output. Practically, these dependencies are assumed negligible based on the Markov cipher assumption, hypothesis of stochastic equivalence and hypothesis of independent round subkeys [96].

Along the years, differential cryptanalysis has proved to be a powerful cryptanalytic technique against symmetric-key primitives and since its introduction, several variants, tweaks and generalizations have been made. Knudsen introduced truncated differential attacks that exploit differentials where only parts of the input and output differences are known [88]. In the same paper and based on Lai's initial consideration [94], Knudsen has also proposed higher-order differentials, which care about multiple differences concurrently by arranging them in distinct levels of recursion. The boomerang attack was later introduced by Wagner, it allows connecting two different differentials over different parts of the cipher that do not overlap in the middle. The boomerang attack was further generalized to the amplified boomerang attack [86] and the rectangle attack [22].

Linear Cryptanalysis: Linear cryptanalysis is another major cryptanalysis technique used against symmetric-key primitives. It was applied on DES by Matsui [108]. In linear cryptanalysis, which is a known plaintext attack, the s-round distinguisher exploited is a linear approximation that holds with probability p:

$$\Gamma_1 \bullet I_1 \oplus \Gamma_{s+1} \bullet I_{s+1} = 0,$$

where $\bullet$ denotes the scalar product over $\mathbb{F}_2$, I_1, I_{s+1} are the input at round 1 and output at round s, respectively, and Γ_1, Γ_{s+1} are called the input and output linear masks. In an ideal block cipher, the above linear approximation holds with probability 0.5. Therefore, the higher the bias $\epsilon = |p - 1/2|$ of this linear approximation is, the more the cipher deviates from an ideal block cipher.

In linear cryptanalysis, the number of known plaintext/ciphertext pairs is inversely proportional to the squared bias, i.e. ϵ^2, of the linear approximation. Similar to differentials, efficient linear approximations are found by searching for linear trails where an s-round linear trail is a sequence of $s+1$ linear masks $(\Gamma_1, \Gamma_2, \ldots, \Gamma_s, \Gamma_{s+1})$. For the full linear trail to hold, each round linear approximation $\Gamma_i \bullet I_i \oplus \Gamma_{i+1} \bullet I_{i+1} = 0$ has to hold with bias ϵ_i. The propagation of the linear approximations is probabilistic through non-linear components and deterministic through linear ones. The bias of an s-round linear trail can be computed as $2^{s-1} \prod_{i=1}^{s} \epsilon_i$, according to the piling-up lemma [108] and under the same assumptions mentioned above in the differential cryptanalysis, where ϵ_i is the bias of the linear approximation at round i. The pair (Γ_1, Γ_{s+1}) defines the so-called linear hull consisting of all linear trails that have the same input and output linear masks. Nyberg showed that the expected squared bias of a linear approximation averaged over all round keys is the sum of the expected squared biases of all relevant linear trails averaged over all round keys [118]. Later, it was proved that for key-alternating ciphers, the expected squared bias of a linear approximation averaged over all round keys is the sum of the squared biases of all relevant linear trails, which are then independent of the key [53].

After its introduction, many extensions and improvements of linear cryptanalysis have been proposed. For example, to reduce the data complexity, chosen-plaintext linear cryptanalysis was proposed [90]. To the same end, Hermelin, Cho, and Nyberg have studied multidimensional linear attacks [74]. Recently, Bogdanov et al. have introduced the concept of zero-correlation attacks [37]. These attacks are based on linear trails with zero bias, i.e., their probabilities are exactly 0.5. A more generalized form of zero-correlation attacks, particularly attacks based on key-invariant biases, was introduced by Bogdanov et al. [34].

Meet-in-the-Middle Attacks: In 1977, Diffie and Hellman were the first to propose the meet-in-the-middle (MitM) attack for the cryptanalysis of DES [61]. Demirci and Selçuk were the first to apply the MitM approach on AES [56] triggering a new line of research, we refer to their attack as the *plain* MitM attack. They have shown that if the input of four AES rounds has just

one active (non-zero difference) byte then the value of each byte of the output can be described as a parameterized function of that active byte. The number of parameters was deduced to be 25 8-bit parameters in [56] and then reduced to 24 8-bit parameters in [57]. The reduction in the number of parameters was possible by noticing that any of the 25 parameters can be taken as a reference for the others. Therefore, by considering the differences of the functions rather than the mere values, only 24 parameters can be used. The main disadvantage of this MitM attack, even with the reduced number of parameters, is the high memory requirement to store a precomputation table of all the sequences resulting from all the possible combinations of these parameters. As such, this attack only works for AES-256 and then a time-memory trade-off is used to extend the attack to AES-192.

At ASIACRYPT 2010, Dunkelman, Keller, and Shamir [67] proposed a couple of new ideas to address the high memory requirement of this MitM attack. First, they showed that the precomputation table does not need to have the whole sequence, just its associated multiset, i.e., the unordered sequence with multiplicity rather than the ordered one. The introduction of the multiset concept reduced the size of the precomputation table by a factor of four. However, the more significant reduction in the size of the precomputation table was due to the second and main idea, which they called the differential enumeration. Differential enumeration reduced the number of parameters that describe the sequence or rather the multiset from 24 bytes to 16 bytes. This is achievable by relying on a low probability truncated differential characteristic where the generated sequence or multiset at its output can only take a restricted number of values. Consequently, the memory requirement has been reduced from 2^{192} to 2^{128}. However, the use of this truncated differential characteristic increases the data complexity as now we have to search through a large amount of input data pairs to find one pair that conforms to the used truncated differential characteristic.

Later on, at Eurocrypt 2013, Derbez, Fouque and Jean showed that it is possible, by borrowing ideas from the rebound attack [111], to enumerate the whole set of sequences more

efficiently [58]. In particular, they showed that using their technique, which they called efficient enumeration, the whole set can take only 2^{80} values instead of 2^{128}. This means that the number of parameters is further reduced to 10 parameters only. The consequences of using the efficient enumeration technique were numerous. Firstly, the attack became feasible on AES-128 and in fact, their attack is considered the most efficient attack on 7-round reduced AES-128. Secondly, the use of a 5-round truncated differential characteristic is feasible which mounts to attacking 9-round reduced AES-256. Thirdly, the memory complexity is no longer the bottleneck of the attack.

Finally, Li, Jia and Wang [99] employed a key-dependent sieve to further reduce the memory complexity of Derbez's attack. This technique helped them present an attack on 9-round reduced AES-192 using a 5-round truncated differential characteristic and an attack on 8-round reduced PRINCE. The key-dependent sieve was later used to mount an attack on 10-round reduced AES-256 [100].

Differential-linear cryptanalysis: The differential and linear attacks have been used jointly for the first time by Langford and Hellman [97]. In the Differential-linear cryptanalysis, the cipher is split into two parts such that for the first part of the cipher, a high-probability truncated differential exists and for the second part, a high-bias linear approximation is found. Langford and Hellman introduced a specific case with a probability-one differential. Following that, Biham et al. [23] generalized the differential-linear cryptanalysis by using a probabilistic truncated differential. Under some assumptions, the resulting linear approximation is estimated to have a bias $\epsilon' = 2p\epsilon^2$, where p is the probability of the differential and ϵ is the bias of the linear approximation. Blondeau et al. [32] have revisited the differential-linear cryptanalysis to apply the theoretical link between linear and differential cryptanalysis [42]. They were able to use a closed expression to accurately express the bias of a differential-linear approximation, with just one assumption, that is the two parts of the cipher are independent. They further introduced a multidimensional generalization of differential-linear cryptanalysis that includes multiple input

differences and multidimensional linear output masks.

Impossible differential cryptanalysis: Biham et al. [21] and Knudsen [87] have independently proposed impossible differential attacks, which is one of the most powerful cryptanalytic techniques. Firstly, we try to find a certain input difference that propagates to a specific output difference with zero probability resulting in an impossible differential distinguisher. In general, the input and output differences can be truncated. Then, after finding the longest possible impossible differential, it is used in a key recovery attack by prepending and/or appending a few additional rounds, which are usually called the analysis rounds. The attack proceeds as follows: first, we collect pairs with certain plaintext and ciphertext differences. Then, we guess some bits of the key material involved in the analysis rounds and if one of the pairs satisfies the input and output differences of the impossible differential under some subkey bits, then these subkey bits must be wrong. Thus, we discard as many wrong keys as possible and do an exhaustive search on the surviving ones along with the rest of the key. The early abort technique [104] allows us to guess the involved key material on steps to discard the undesired pairs as early as possible and therefore reduce the time complexity of the attack.

Structural cryptanalysis. While statistical cryptanalysis exploits low-level statistical behavior of the block cipher components, structural cryptanalysis relies on its high level properties. Below, we give two examples of structural attacks.

Multiset attacks: Multiset attacks are chosen-plaintext attacks, which are comparable to differential cryptanalysis. In Multiset attacks, the adversary tracks the deterministic propagation of a chosen set of plaintexts (multisets) and their properties through the cipher, where a multiset is a set where each element can appear several times. Thus, the exploited distinguisher in these attacks has a probability of one. Examples of multiset attacks include the square attack [50] and the integral cryptanalysis [93], which was quite recently generalized by the division property [140].

Slide attacks and Reflection attacks: While slide attacks, introduced by Biryukov and Wagner [27, 28], exploit self-similarity properties of the block cipher round functions, reflection attacks, proposed by Kara and Manap [82, 83], exploit self-similarity between its encryption and decryption algorithms. Slide attacks are so powerful that, if they exist, they would be independent of the number of rounds of the block cipher, i.e., they can attack any number of rounds. However, the countermeasure against these attacks is quite simple by introducing round constants to break any self-similarity.

Chapter 3

MILP Modeling for (Large) S-boxes to Optimize Probability of Differential Characteristics

Over the past few years, Mixed Integer Linear Programing (MILP) has been successfully employed to efficiently find the best differential characteristic, linear approximation, longest impossible differential, zero-correlation linear approximation and division trail for many symmetric-key primitives. In the existing approach, the linear inequalities used to restrict the input and output variables of an n-bit S-box to the region of feasible solutions can be generated by using either SageMath or multiple logical condition operations. Both methods become computationally infeasible when the size of the S-box input exceeds 5-bits.

In this chapter, we address this limitation and propose an efficient way to generate the linear inequalities representing the differential propagation through large S-boxes (6-bit and upwards). We start by separating the DDT entries of the S-box by their values. Then, we assign a Boolean function to each distinct value and transform each Boolean function to its minimized product-of-sum form, which can be efficiently represented by a set of linear inequalities. Our approach allows us to precisely evaluate the probability of differential characteristics. We have applied

our proposed approach on two of the efficient AES-round function based constructions proposed in FSE 2016 by Jean and Nikolić and managed to improve the lower bound on the number of the active S-boxes in one construction and the upper bound on the differential characteristic of the other. Although we have applied our approach in the context of differential cryptanalysis, its application can easily be extended to linear cryptanalysis.

Parts of this chapter have been published in [3].

3.1 Introduction

The use of Mixed Integer Linear Programming (MILP) as a supporting tool in the symmetric-key cryptography has started by Mouha et al. [113] and Wu and Wang [152]. They have proposed two slightly different approaches to model the problem of finding a lower bound on the number of active S-boxes for both differential and linear cryptanalysis as an MILP problem that can be solved by any MILP solver such as Gurobi Optimizer [71], and CPLEX Optimization Studio [75]. Such a lower bound and the maximum differential (linear) probability of the S-box derive an upper bound on the probability of the best differential characteristic (linear approximation). This helps the designer of symmetric-key primitives prove its resistance against differential (linear) cryptanalysis after a given number of rounds.

The use of MILP in symmetric-key cryptography has been amplified in the past few years. It was used by Sun et al. to find the best differential characteristic in a number of bit-oriented SPN block ciphers [134] in the single-key and related-key settings. They have used a heuristic approach and therefore the automatically found differential characteristic has to be verified manually as it might be invalid. Shortly afterwards, Sun et al. proposed a methodology to exactly represent the differential propagation through an S-box [133]. Their methodology was computationally feasible when the size of the S-box is small, i.e., 5-bit or less. Then, Fu et al. proposed an MILP-based method to search for the best differential and linear trails in Addition

Rotation XOR (ARX) block ciphers under the assumption of independent inputs to both the modular addition and the different rounds [69].

Recently, Sasaki and Todo presented an MILP-based tool to automatically search for the longest impossible differential in SPN based block ciphers [127]. They have pointed out the inability of the current approaches to efficiently represent large S-boxes and suggested the use of, what they have called, the arbitrary S-box mode to represent the differential propagation of the 8-bit S-boxes DDT. Independently, Cui et al. proposed a similar tool to search for impossible differentials and zero-correlation linear approximations with emphasis on ARX block ciphers [49].

MILP usage was not limited to differential and linear cryptanalysis only and was extended to Integral cryptanalysis. Xiang et al. [153] have proposed an MILP-method to find integral distinguishers based on the division property [140] and applied it on six lightweight block ciphers. Soon after, their approach was extended to primitives with non-bit permutation linear layers and ARX based primitives by Sun et al. [130, 131]. However, the solutions of Sun et al. were found to encompass some infeasible division trails, which could affect the search results and yield shorter integral distinguisher as was shown in [154].

In this chapter, we present a new MILP modeling that can be applied to the DDT of (large) S-boxes. This modeling permits the utilization of the MILP approach on block ciphers that employ 8-bit S-boxes. First, we separate the DDT according to each distinct probability value and assign a Boolean function to each of these values. The truth tables of these Boolean functions are built considering that their inputs are the input and output difference bits and each function evaluates to 1 when its corresponding probability value occurs for that input-output difference pair, and 0 otherwise. Then, these truth tables are converted to their minimum Product of Sum (PoS) forms and each term in the PoS is converted to a linear inequality. The special case of zero input difference being propagated to zero output difference is handled separately.

Out of the seven constructions (C1 to C7) proposed in [77], we applied our modeling approach on two arbitrarily selected constructions, i.e., C1 and C5. The designers showed that the minimum number of active S-boxes is 22 for both constructions, which ensures that the upper bound on the probability is 2^{-132}. Regarding C1, we show that the best probability is slightly smaller, i.e., 2^{-134}. Regarding C5, we show that none of the truncated differentials that have 22 or even 23 active S-boxes can be instantiated. Hence, we improve the lower bound on the number of active S-boxes for C5 by two.

We emphasize that while our focus is on differential cryptanalysis, the proposed method for efficiently modeling large S-boxes can directly be used in other contexts such as linear cryptanalysis, impossible differential attack, zero-correlation attack, and integral attack.

The remainder of this chapter is organized as follows. In Sect. 3.2, we explain how to model the problem of finding the best differential characteristic for an SPN block cipher using MILP and list the limitation of this approach. In Sect. 3.3, we present a new modeling of the DDT of (large) S-boxes to efficiently optimize the probability of differential characteristics. The proposed modeling is applied to two of the AES-round based constructions in Sect. 3.4. Finally, the chapter is summarized in Sect. 3.5.

3.2 Existing MILP Modeling of an SPN block cipher

To automatically find the best differential characteristic of a(n SPN) block cipher, Sun et al. [133, 134] proposed describing the differential behavior of a block cipher as an MILP model. The feasible region of such an MILP model should contain only the set of all valid differential characteristics of the considered block cipher. To build such MILP model, binary variables x_i are assigned to each input and output difference bit of every operation used in the cipher, i.e., when $x_i = 1$, this bit has a nonzero difference, and $x_i = 0$ denotes no difference. The objective function of this model would be to maximize the probabilities associated with the differential

propagation through the different operations. If we use the negative of log base 2 of these probabilities, then the objective function is to minimize these log values. As the difference propagates deterministically through linear operations, the objective function involves the probabilities of the differential propagation through non-linear operations. Then, the differential propagation through each operation is described as a set of linear inequalities that are used as constraints in this MILP model. For example, the differential propagation through the basic XOR operation of two variables ($x_0 \oplus x_1 = y_0$) can be described by the following linear inequalities:

$$x_0 + x_1 - y_0 \geq 0, \quad x_0 - x_1 + y_0 \geq 0, \quad -x_0 + x_1 + y_0 \geq 0, \quad -x_0 - x_1 - y_0 + 2 \geq 0 \quad (3.1)$$

When the block cipher employs a generic linear transformation, it can be represented as a binary matrix and then converted to XOR operations involving multiple bits. When the block cipher employs a permutation, similar to the ShiftRows operation utilized by AES, it can be handled by simply permuting the binary bits of the internal state. In general, the modeling of the differential propagation through linear operations is done in a straightforward manner and the challenge is to model the differential propagation, i.e., the DDT of an S-box. To this end, two approaches were presented: *logical condition modeling* and *H-representation of the convex hull*.

Logical condition modeling. In this approach, general MILP ideas to model logical conditions as linear inequalities are borrowed to describe the conditional differential properties of an S-box. For example, it was observed in [137], that if the output difference of the S-box employed by the PRESENT block cipher [36] is (0101), then the least significant bit of the input difference is always 0. This logical differential property can be modeled using the following linear inequality:

$$-x_3 + y_0 - y_1 + y_2 - y_3 + 2 \geq 0.$$

where (x_0, x_1, x_2, x_3) and (y_0, y_1, y_2, y_3) are the MILP binary variables representing the input and output difference bits, respectively, and x_3 and y_3 are the least significant bits.

H-representation of the convex hull. In this approach, the n input difference bits of every valid differential propagation of the DDT of an S-box are concatenated with the corresponding m output difference bits to form $n + m$ bit vectors. The H-representation of the convex hull, which is the smallest convex set containing these $n + m$ vectors is calculated using SageMath [60]. There might be redundancy in the linear inequalities returned by SageMath, hence, in order to reduce this set of linear inequalities, a greedy algorithm was developed [134]. The probability of each valid input-output difference pattern can be encoded into p bits and appended to the $n+m$ bit vectors before using SageMath to generate the H-representation of their convex hull. This way, an exact upper bound on the probability of the differential characteristic can be obtained.

One major shortcoming of the current approaches to model the DDT of an S-box is its applicability to large S-boxes, i.e., 8-bit S-boxes. This has been pointed out several times, for example, Sun et al. wrote that "To the best of our knowledge, the MILP approach is unable to search for actual differential characteristics of ciphers with 8×8 S-boxes" [132]. Sasaki and Todo wrote "MILP requires too many inequalities to represent differential propagations in DDT of 8-bit S-boxes" [127]. In the next section, we propose a new modeling approach to address this limitation.

3.3 New MILP Modeling to Optimize Probability of Differential Characteristics

In our new model, we separate the DDT entries of an S-box according to the values of their probabilities. Then, we treat each probability value as a Boolean function whose inputs are

the input and output difference bits, i.e., this Boolean function evaluates to 1 when the input difference goes to the output difference with that probability and 0, otherwise. This results in a number of Boolean functions that equals the number of distinct probability values in the DDT. Afterwards, each Boolean function is transformed to its reduced PoS form, which, in turn, is represented by a set of linear inequalities. The sets of linear inequalities representing all the Boolean functions symbolizing all the probability values of the DDT are combined together and integrated into an MILP model that represents the differential propagation through the S-box.

The deterministic transition of zero input difference going to zero output difference is handled a bit different. Instead of having a Boolean function dedicated to this transition, we assign a binary variable that can be thought of as a flag. When this binary variable is off, i.e. set to zero, the S-box is not active and the input and output difference bits are all set to zero. When the flag is on, the S-box is active and one probability value along with input and output differences must be chosen.

input	0	1	2	3	4	5	6	7
output	5	3	4	6	2	7	0	1

Table 3.1: An arbitrary 3-bit S-box Representation (in decimal)

To illustrate our approach, we apply its steps on the arbitrary 3-bit S-box in Table 3.1 and whose DDT is given in Table 3.2. Other than the deterministic zero input to zero output transition, there are two distinct probability values: 2^{-1} and 2^{-2}. Each of these values will be assigned a Boolean function symbolized by, e.g., p_0 and p_1. Then, the truth tables of these Boolean functions are created. In Table 3.3, we show the entries of these truth tables where either p_0 or p_1 equals one and in all the omitted entries both p_0 and p_1 equal zero. Then, each Boolean function is, individually, transformed to its reduced PoS form[1]. For example, the PoS of p_0 is as follows:

[1]Logic Friday [102] is an example of an off-the-shelf software that can be used to minimize the PoS of a Boolean function.

$$p_0 = (\overline{y_1} \vee y_2) \wedge (\overline{x_1} \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (y_1 \vee \overline{y_2}) \wedge (y_0 \vee y_2)$$
$$\wedge (x_0 \vee y_2) \wedge (x_2 \vee y_0) \wedge (x_0 \vee x_2) \wedge (\overline{x_0} \vee \overline{x_2} \vee \overline{y_0} \vee \overline{y_2}),$$

where $\overline{x_i}$ and $\overline{y_i}$ denote the negation of x_i and y_i, respectively, i.e., $\overline{x_i} = x_i \oplus 1$ and $\overline{y_i} = y_i \oplus 1$, and $\wedge$, $\vee$ represent the logical AND and OR operations, respectively.

Input Difference (x_0,x_1,x_2)	Output Difference (y_0,y_1,y_2)							
	0	1	2	3	4	5	6	7
0	1	0	0	0	0	0	0	0
1	0	2^{-2}	2^{-2}	0	0	2^{-2}	2^{-2}	0
2	0	2^{-2}	2^{-2}	0	0	2^{-2}	2^{-2}	0
3	0	0	0	2^{-1}	0	0	0	2^{-1}
4	0	0	0	0	2^{-1}	0	0	2^{-1}
5	0	2^{-2}	2^{-2}	0	0	2^{-2}	2^{-2}	0
6	0	2^{-2}	2^{-2}	0	0	2^{-2}	2^{-2}	0
7	0	0	0	2^{-1}	2^{-1}	0	0	0

Table 3.2: The DDT of the 3-bit S-box given in Table 3.1

The above equation means that for p_0 to be one, all the terms must equal 1, for instance from the first term either $y_1 = 0$ or $y_2 = 1$ and from the last one either x_0, x_2, y_0 or y_2 is 0 and similarly for all the other terms.

The next step is to represent these terms as linear constraints and this can be done by using either implied constraints if the MILP solver has such constraint built-in or model them as conditional constraints. The built-in implied constraint representing the last term would be:

$$p_0 = 1 \Rightarrow (1 - x_0) + (1 - x_2) + (1 - y_0) + (1 - y_2) \geq 1$$
$$p_0 = 1 \Rightarrow -x_0 - x_2 - y_0 - y_2 \geq -3$$

x_0	x_1	x_2	y_0	y_1	y_2	p_0	p_1
0	0	1	0	0	1	0	1
0	0	1	0	1	0	0	1
0	0	1	1	0	1	0	1
0	0	1	1	1	0	0	1
0	1	0	0	0	1	0	1
0	1	0	0	1	0	0	1
0	1	0	1	0	1	0	1
0	1	0	1	1	0	0	1
0	1	1	0	1	1	1	0
0	1	1	1	1	1	1	0
1	0	0	1	0	0	1	0
1	0	0	1	1	1	1	0
1	0	1	0	0	1	0	1
1	0	1	0	1	0	0	1
1	0	1	1	0	1	0	1
1	0	1	1	1	0	0	1
1	1	0	0	0	1	0	1
1	1	0	0	1	0	0	1
1	1	0	1	0	1	0	1
1	1	0	1	1	0	0	1
1	1	1	0	1	1	1	0
1	1	1	1	0	0	1	0

Table 3.3: Binary tables for p_0 and p_1 for the 3-bit S-box DDT

and modeling it as a conditional constraint involve choosing a sufficiently large upper bound (M) on those constraints (See section 7.4 in [29]). In our exemplary case, M would be chosen to be six which is the number of input and output difference bits. Thus, the conditional constraint of the same last term would be:

$$(1 - x_0) + (1 - x_2) + (1 - y_0) + (1 - y_2) - 6p_0 + 6 \geq 1$$

$$-x_0 - x_2 - y_0 - y_2 - 6p_0 \geq -9$$

In this last equation, if p_0 is zero then x_2, x_0, y_2 and y_0 can be chosen freely while if p_0 is 1 then this imposes the restriction that at least one of them must be 0. All the other terms in p_0 and p_1 are represented by linear constraints in a similar manner. To handle the special

case of zero input difference going to zero output difference, we add a binary variable, e.g., f which, as stated above, would be a flag to indicate whether the S-box is active or not. The linear constraints for this would be as follows:

$$\begin{aligned} x_0 + x_1 + x_2 + y_0 + y_1 + y_2 - f &\geq 0 \\ f - x_i &\geq 0 \\ f - y_i &\geq 0, \\ \textit{where } 0 \leq i &\leq 2 \end{aligned}$$

To indicate that only one of the probabilities along with the flag must be set we add the following constraint:

$$p_0 + p_1 - f = 0$$

Finally, the objective function of just one S-box will take the form:

$$\text{Minimize: } p_0 + 2p_1$$

3.4 Application to AES-Based Constructions

The well-understood security of AES and its efficient implementation have made it an appealing building block to construct other symmetric-key primitives such as MACs and AE schemes. For the purpose of being more efficient and without sacrificing security, some primitives have been built using a 4-round reduced version of AES making advantage of its wide-trail strategy [53] which provides an adequate level of security for that number of rounds. However, and due to a much more complicated security analysis, fewer schemes have been constructed using less than four rounds.

In their FSE 2016 paper, Jean and Nikolić have investigated the limits of building efficient AES-based constructions without affecting their security [77]. They made use of the AES-NI dedicated set of instructions available on almost any common processor nowadays with focus on the aesenc instruction which executes one full AES round. To make sure that their schemes have high efficiency, they have carefully studied and subsequently chosen the internal state size and the number of aesenc calls per step and which state blocks should the aesenc instruction be applied to.

On the other hand, they set the only security requirement of their constructions to be resistant against internal collisions. Internal collisions in cryptographic primitives such as MACs and AEs are normally based on a differential characteristic with high probability that starts and ends with zero difference internal states after introducing some intermediate states differences using the message blocks. As the maximum differential probability of the AES S-box is 2^{-6} and the highest differential that can be used in an internal collision must avoid allowing a probability higher than 2^{-128} (when the key size is 128 bits), this yields a lower bound on the number of active S-boxes of $\lceil 128/6 \rceil = 22$.

To search for secure constructions whose best differential has a minimum of 22 active S-boxes, Jean and Nikolić have used MILP as was done in [113]. They have shown seven secure constructions that provide a good trade-off between state size and efficiency (see Figures 5-11 in [77]). The state size of these constructions ranges from 6-9 and 12 128-bit blocks. Two constructions process three message blocks per step, i.e., an iteration of their designs, while the others process two message blocks per step. In all these constructions, the AES round function is applied on particular state blocks only once per step. This was found and proven to be more efficient than cascading r AES round function calls.

However, Jean and Nikolić have noted the limitation of MILP-based search as follows:

Limitations. *"Despite providing a simple and efficient way of finding differential*

characteristics, MILP only yields upper bounds on the actual probabilities of the differential characteristics as, theoretically, they can be impossible. We emphasize that this does not relate to impossible differential characteristic, but to the fact that partially undetermined behavior of the XOR operation (mentioned before) may result in inconsistent systems that produce truncated differential characteristics, which are impossible to instantiate with actual differences. Fortunately, while a cryptanalyst should ensure the validity of the produced characteristics, we, as designers, only need to confirm that the upper bound on the probability of the best differential characteristic is sufficiently low."

This has motivated us to attempt instantiate two of their constructions using our new model. To this end, we adopt the following two-stage search approach, similar to the one introduced by Sun et al. [132] for constraint programming:

Stage 1. Using MILP as was done in [113], we search for all the truncated differentials that have the minimum (or a predefined) number of active S-boxes. It is to be noted that it is not known yet whether these truncated differentials can be instantiated with actual differences or not.

Stage 2. We try to instantiate each truncated differential by using our new proposed model while considering the positions of the active and non-active S-boxes.

3.4.1 Results on the Fifth Construction (C5)

C5 processes two message blocks per step with an internal state of seven 128-bit blocks and the AES round function call is applied to five out of these seven blocks. The designers reported that the minimum number of active S-boxes for this construction is 22 but without stating the number of steps needed to reach it. Therefore, our first step was to find the truncated differentials that achieve that minimum number of active S-boxes and in how many steps. We have found that

there are only four truncated differentials that achieve the minimum number of active S-boxes after four steps. On the chance that there might be other truncated differential characteristics that have 22 active S-boxes but for larger number of steps. We ran the MILP model for five, six, seven and eight steps and the number of differential characteristics did not change. All these differentials were found to be infeasible. One of which is depicted in Figure 3.2. Verifying it manually, for $BX4$ and $DX4$ to be of zero difference, bytes 12-15 of $BX3$, $CX3$, and $M1\_3$ must be equal. As bytes 12-15 in $BZ2$ are zero, this means that these bytes also equal bytes 12-15 in $M1\_2$, i.e., bytes 12-15 in $BX3$, $CX3$, $M1\_2$, and $M1\_3$ are equal. As $AZ2$ is all zeros, this means that bytes $BX2$ must be inactive which contradicts the truncated differential. Similar reasoning applies to the other three truncated differentials with 22 active S-boxes.

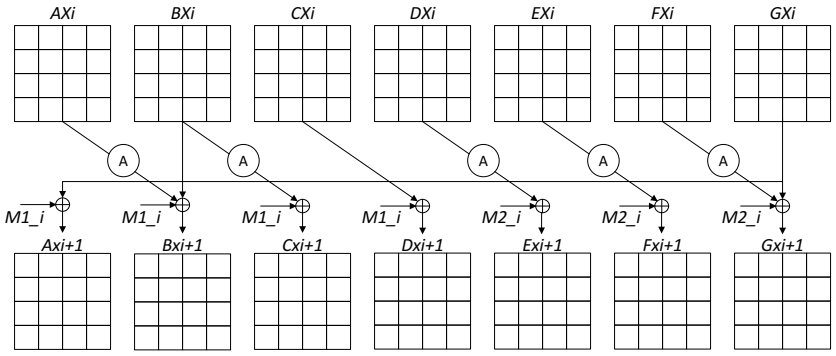

Figure 3.1: C5 construction. 'A' represents one AES round function.

Then, we increased the number of active S-boxes and found 580 truncated differentials that have 23 active S-boxes. Again, we verified that with higher number of steps, the number of truncated differentials with 23 actives S-boxes does not change. All these truncated differentials were found to be infeasible by the solver. One of these differentials is depicted in Figure 3.3. We found that it is invalid because of the reasoning mentioned above, i.e., for $BX4$ and $DX4$ to be of zero difference, bytes 13-15 of $BX2$ should be inactive contradicting the differential characteristic. Moreover, for $GX4$ to be of no difference, it requires that $MC[0, a, 0, 0]$ from $FZ2$ to be equal to $MC[b, 0, 0, 0]$ from $FZ3$ and this is not possible because there are no non-zero differences at different positions that would yield the same output after the AES MixColumn,

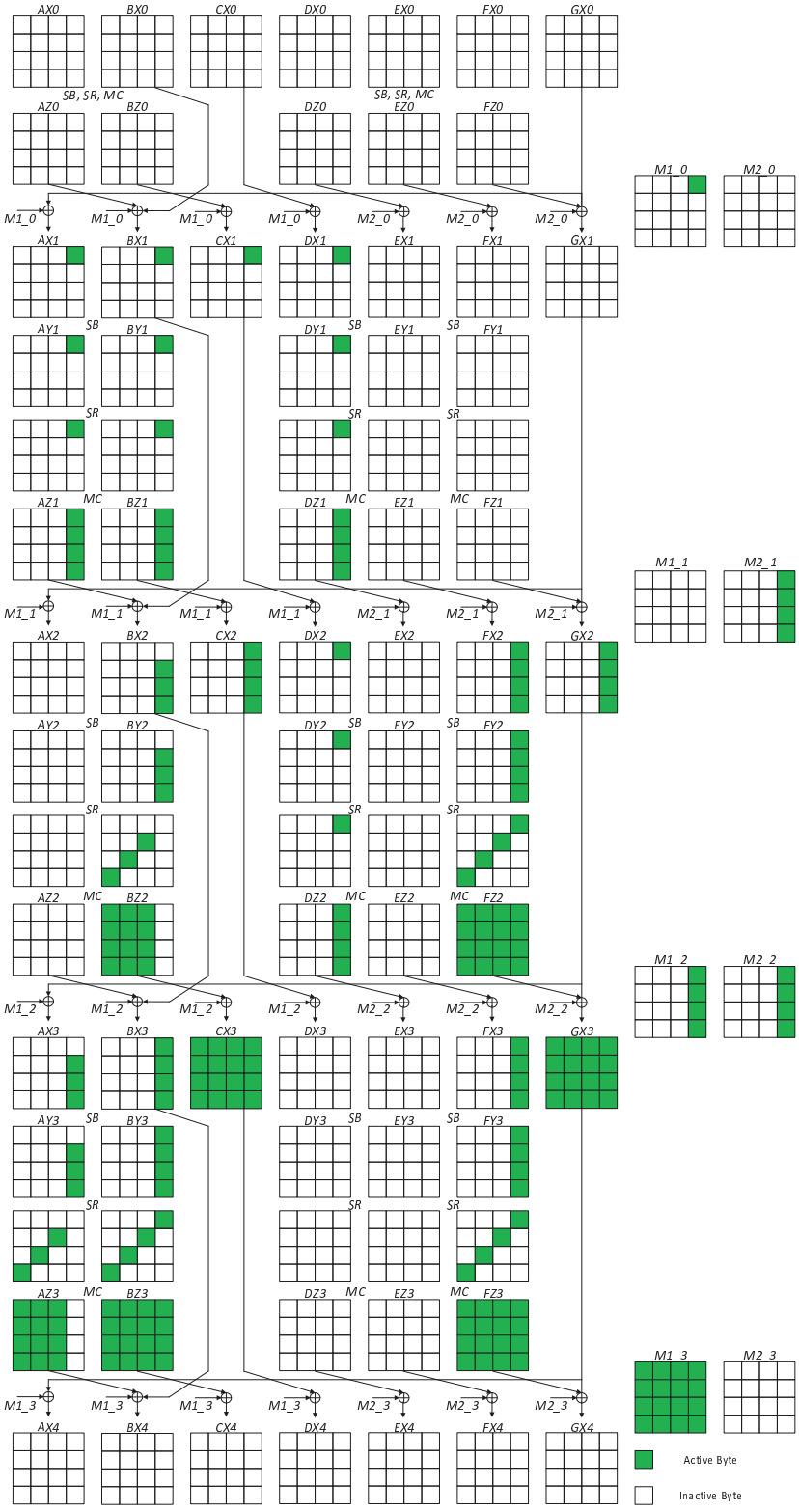

Figure 3.2: Invalid differential characteristic for C5 with 22 active S-boxes

MC, operation.

This concludes that the lower bound on the number of active S-boxes for C5 is 24, and not 22 as early estimated by the designers.

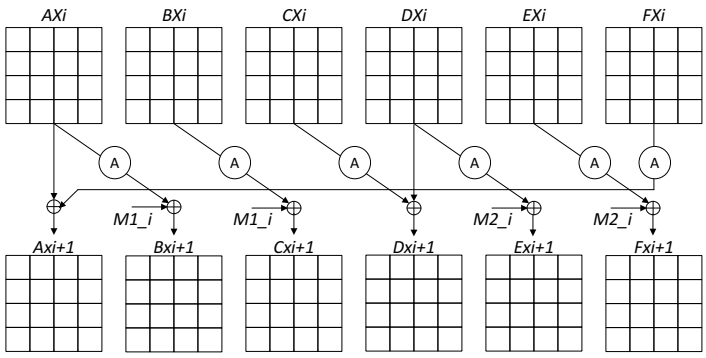

Figure 3.4: C1 construction. 'A' represents one AES round function.

3.4.2 Results on the First Construction (C1)

C1 processes two message blocks per step with an internal state of 6 128-bit blocks and the AES round function call is applied to all these 6 blocks. The minimum number of active S-boxes reported by the designers is 22 with no stated number of steps. 256 truncated differential characteristics were found to achieve that minimum number of active S-boxes after three steps. We checked up to seven steps and the number of truncated differentials remained constant. These 256 differential characteristics are formed by having one active byte in the first block of both messages. We have applied our method on all of these differential characteristics and 252 were found by the solver to be infeasible. Figure 3.5 depicts one of these invalid differential characteristics. We have verified manually that this characteristic is indeed invalid. As shown in Figure 3.5, for $AX3$ to be of zero difference, this means that $FZ2$ must equal $AX2$ and as $AX1$ is all zeros, $AX2$ equals $FZ1$. So, for $AX3$ to be of zero difference $FZ1 = MC[0, a, 0, 0]$ must equal $FZ2 = MC[b, 0, 0, 0]$ and this is not possible as explained above. Following the same logic, the truncated differential can be valid if and only if the active bytes are at the same position and that position is not impacted by the shift row operation along the different steps,

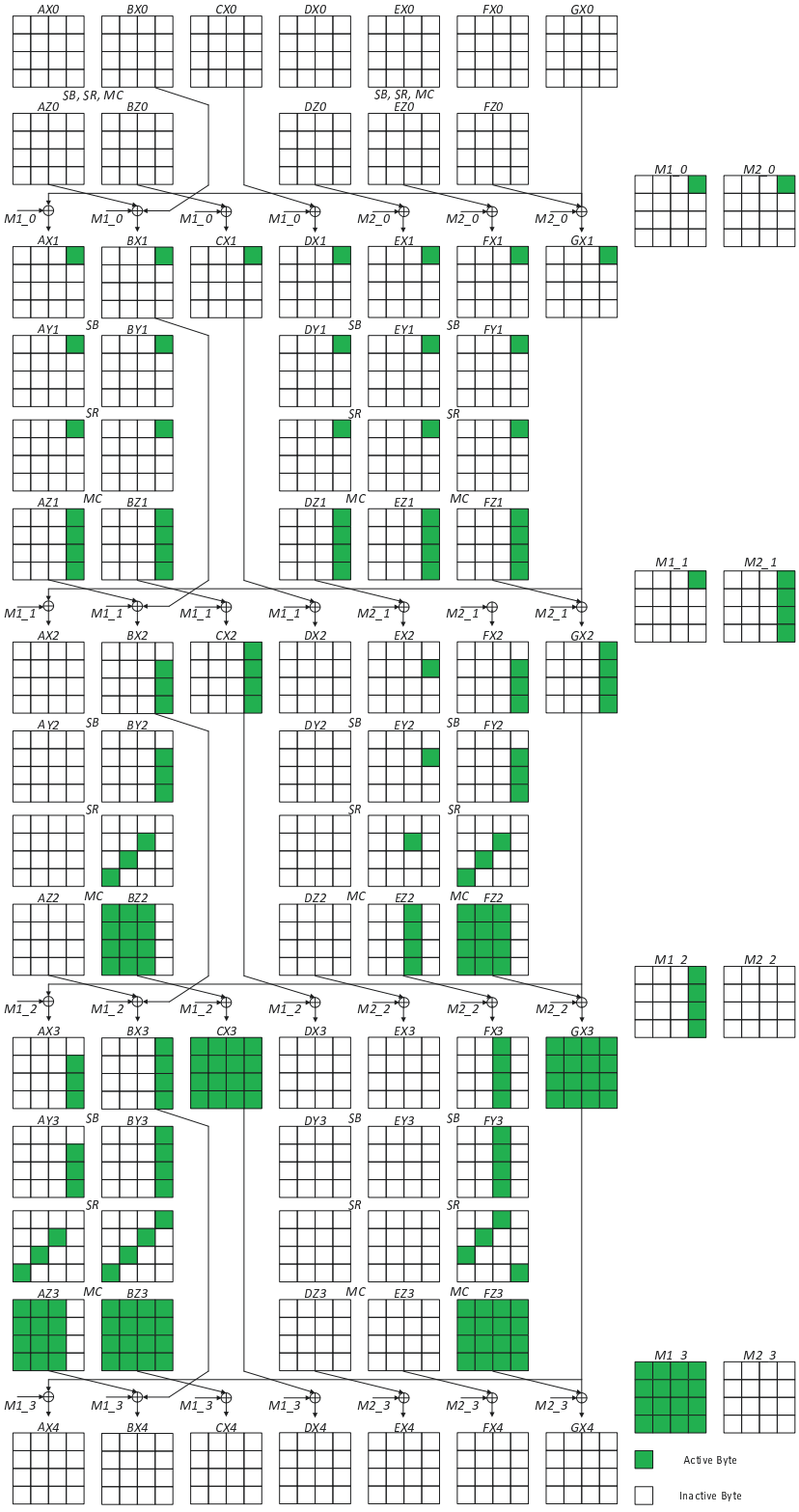

Figure 3.3: Invalid differential characteristic for C5 with 23 active S-boxes

i.e., the active bytes of the first block of the two messages are at either (0, 0), (4, 4), (8, 8) or (12, 12), i.e., the four truncated differentials found to be feasible by the solver.

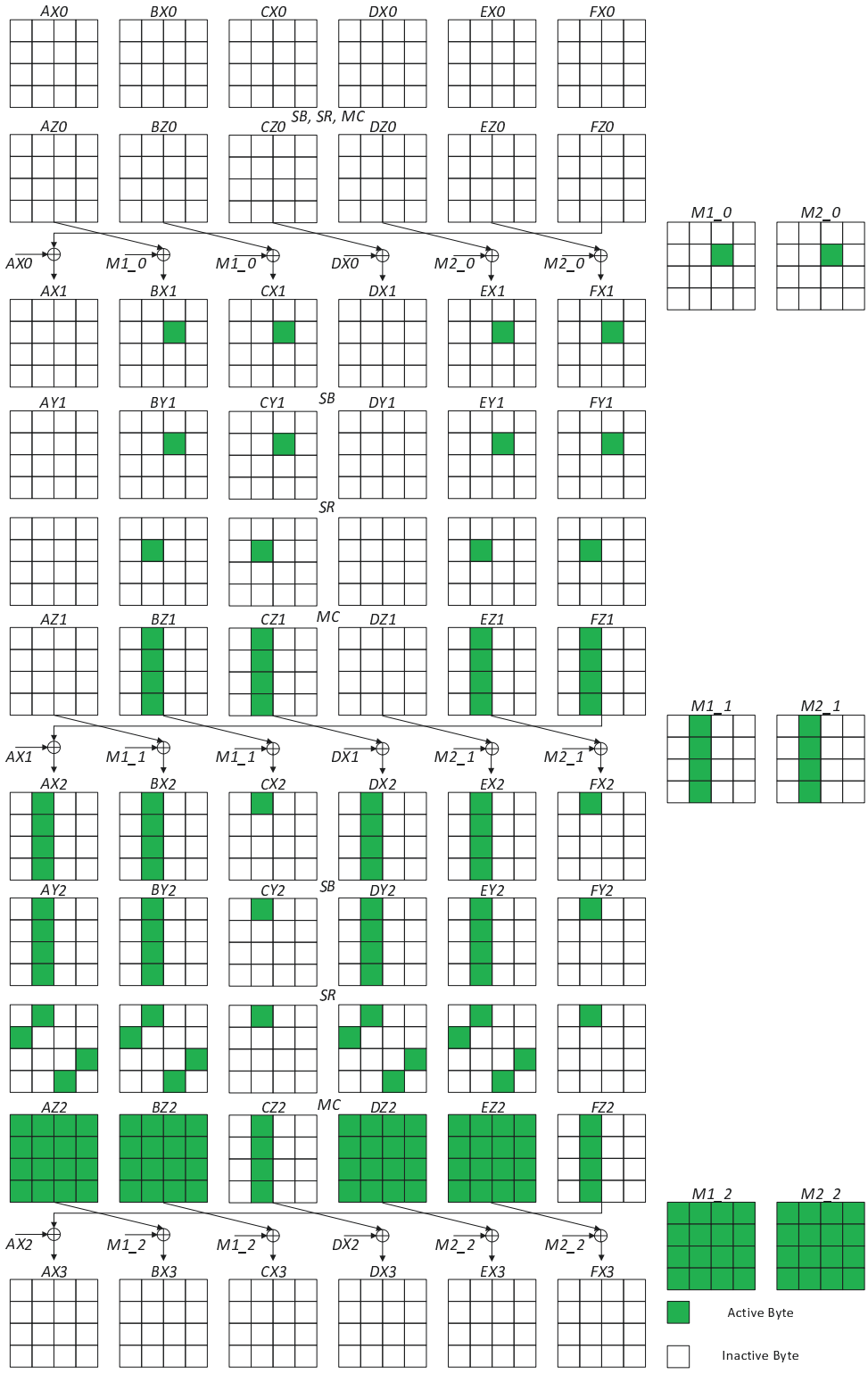

Figure 3.5: Invalid differential characteristic for C1

As the S-box is the same for every byte of the state and as the MC input will always have the same format, the probability of these four differentials will be the same and the actual differences will just be a permutation of each other. We have tried to instantiate one of these four differential characteristics and it was found that the best probability is 2^{-134} instead of 2^{-132} which means that there are two S-boxes that cannot be bypassed by the maximum probability of 2^{-6}. These are the S-boxes at byte 8 of $AX2$ and $EX2$ (highlighted in yellow in Figure 3.6 and the actual difference values are listed in Table 3.4). Looking for the reason, we have found that for the path to be valid, the input of the S-boxes at byte 8 of $AX2$ and $BX2$ (resp. $EX2$ and $FX2$) must be different and the output must be the same, i.e., $S(a) = S(b)$ where are a, b are two distinct non-zero differences. We have searched through the DDT of the AES S-box and found that the maximum probability that would fulfill this condition is 2^{-13} , one S-box is activated with 2^{-6} while the other is activated with 2^{-7}. This means that the probability found by the MILP solver is the highest probability. Therefore, we conclude that the best differential characteristic of this construction has a probability of 2^{-134}.

3.5 Summary

In this chapter, we presented a new MILP modeling of the DDT of (large) S-boxes. At first, the DDT is split into multiple tables according to the probability value. A Boolean function is assigned to each value and then using some of the off-the-shelf software, its truth table is converted to the minimum product of sums form. The latter is represented as a set of linear inequalities that can efficiently describe the differential propagation through large S-boxes, i.e., 8-bit S-boxes. With the proposed modeling, we evaluated the upper bound on differential characteristics of two AES-round based constructions. We improved the lower number on active S-boxes for one construction and improved the upper bound on the differential probability of the other. Lastly, it is worth noting that the proposed modeling can be applied in other cryptanalysis techniques such as linear cryptanalysis.

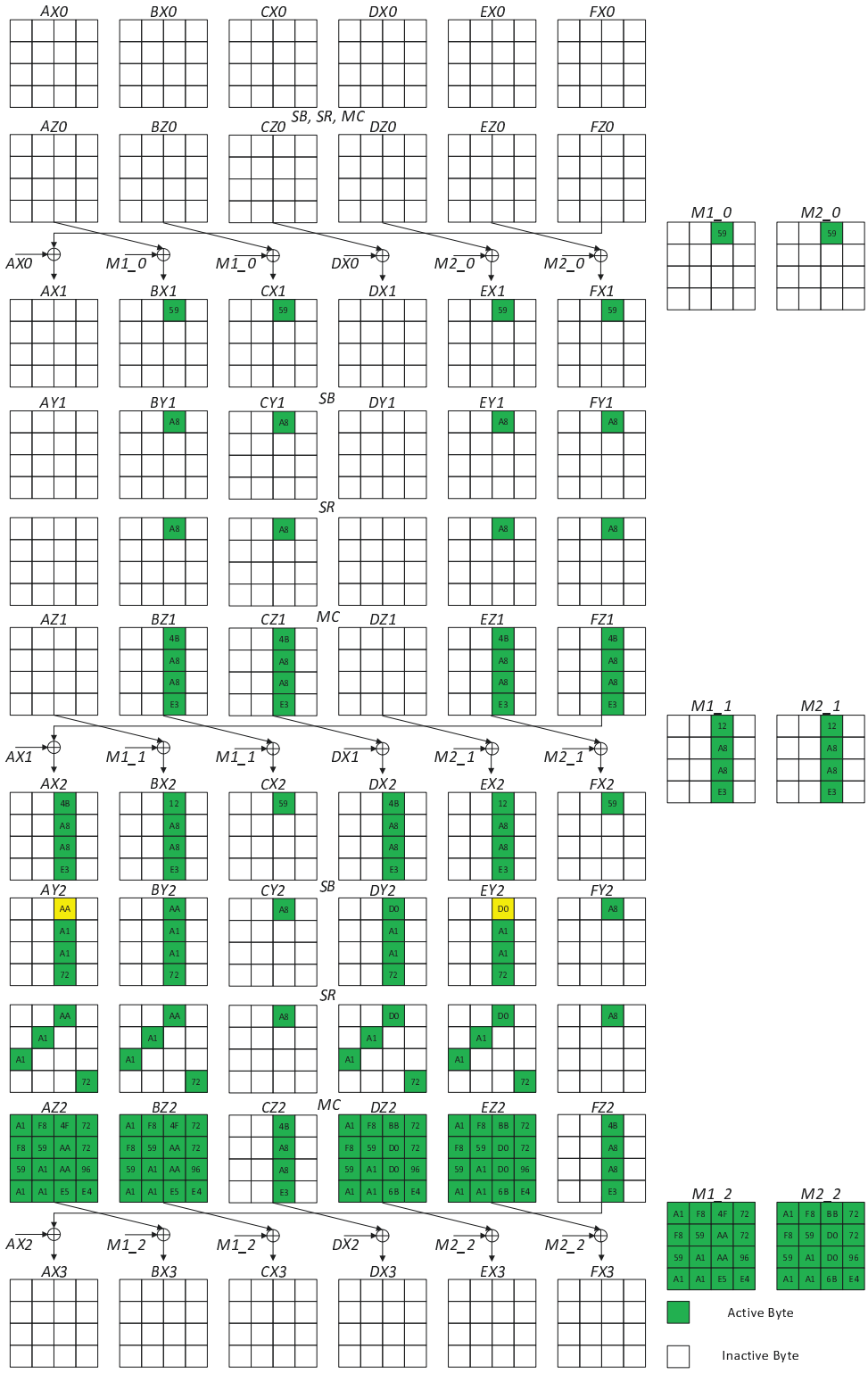

Figure 3.6: Valid differential characteristic for C1

State	Values	Probability	State	Values	Probability
$M1\_0$ $M2\_0$	00 00 59 00 00 00 00 00 00 00 00 00 00 00 00 00	N/A	BX_1 CX_1 EX_1 FX_1	00 00 59 00 00 00 00 00 00 00 00 00 00 00 00 00	N/A
BY_1 CY_1 EY_1 FY_1	00 00 A8 00 00 00 00 00 00 00 00 00 00 00 00 00	[1 1 -6 1] [1 1 1 1] [1 1 1 1] [1 1 1 1]	BZ_1 CZ_1 EZ_1 FZ_1	00 00 4B 00 00 00 A8 00 00 00 A8 00 00 00 E3 00	N/A
$M1\_1$ $M2\_1$	00 00 12 00 00 00 A8 00 00 00 A8 00 00 00 E3 00	N/A	AX_2 DX_2	00 00 4B 00 00 00 A8 00 00 00 A8 00 00 00 E3 00	N/A
BX_2 EX_2	00 00 12 00 00 00 A8 00 00 00 A8 00 00 00 E3 00	N/A	CX_2 FX_2	00 00 59 00 00 00 00 00 00 00 00 00 00 00 00 00	N/A
AY_2	00 00 AA 00 00 00 A1 00 00 00 A1 00 00 00 72 00	[1 1 -7 1] [1 1 -6 1] [1 1 -6 1] [1 1 -6 1]	BY_2	00 00 AA 00 00 00 A1 00 00 00 A1 00 00 00 72 00	[1 1 -6 1] [1 1 -6 1] [1 1 -6 1] [1 1 -6 1]
CY_2 FY_2	00 00 A8 00 00 00 00 00 00 00 00 00 00 00 00 00	[1 1 -6 1] [1 1 1 1] [1 1 1 1] [1 1 1 1]	DY_2	00 00 D0 00 00 00 A1 00 00 00 A1 00 00 00 72 00	[1 1 -6 1] [1 1 -6 1] [1 1 -6 1] [1 1 -6 1]
EY_2	00 00 D0 00 00 00 A1 00 00 00 A1 00 00 00 72 00	[1 1 -7 1] [1 1 -6 1] [1 1 -6 1] [1 1 -6 1]	AZ_2 BZ_2	A1 F8 4F 72 F8 59 AA 72 59 A1 AA 96 A1 A1 E5 E4	N/A
CZ_2 FZ_2	00 00 4B 00 00 00 A8 00 00 00 A8 00 00 00 E3 00	N/A	DZ_2 EZ_2	A1 F8 BB 72 F8 59 D0 72 59 A1 D0 96 A1 A1 6B E4	N/A
$M1\_2$	A1 F8 4F 72 F8 59 AA 72 59 A1 AA 96 A1 A1 E5 E4	N/A	$M2\_2$	A1 F8 BB 72 F8 59 D0 72 59 A1 D0 96 A1 A1 6B E4	N/A

Table 3.4: 3-step differential characteristic for C1 with probability 2^{-134}. Differences are represented by hexadecimal numbers. "Probability" shows logarithm of the probability of the S-box differential transition in the corresponding byte of a particular state.

Chapter 4

Improved Key Recovery Attack on Round-reduced Hierocrypt-L1 in the Single-Key Setting

In this chapter, we analyze the security of the Hierocrypt-L1 block cipher against MitM attack. Hierocrypt-L1 is a 64-bit block cipher with a 128-bit key. It was selected among the Japanese e-Government 2003 recommended ciphers list and has been reselected in the 2013 candidate recommended ciphers list. In this chapter, we cryptanalyze Hierocrypt-L1 in the single-key setting. In particular, we construct a five S-box layer distinguisher that we utilize to launch a MitM attack on eight S-box layer round-reduced Hierocrypt-L1 using the differential enumeration technique. Our attack allows us to recover the master key with data complexity of 2^{49} chosen plaintexts, time complexity of $2^{114.8}$ eight S-box layers Hierocrypt-L1 encryptions and memory complexity of 2^{106} 64-bit blocks. This is the first cryptanalysis result that reaches eight S-box layers of Hierocrypt-L1 in the single-key setting.

Parts of this chapter have been published in [4].

4.1 Introduction

Hierocrypt-L1 (HC-L1) [48, 120, 148], designed by Toshiba Corporation in 2000, is a 64-bit block cipher with a 128-bit key. The cipher employs a nested SPN structure [120], where each S-box in the higher SPN level encompasses a lower-level SPN structure. It was submitted to the New European Schemes for Signatures, Integrity, and Encryption (NESSIE) project [116]. In 2003, it was selected as one of the Japanese e-Government recommended ciphers [46], and its security was reaffirmed by CRYPTREC in 2013 where it was included in the candidate recommended ciphers list [45].

The best-known attack on HC-L1 in the single-key setting is the square attack on seven S-box layers which was proposed by the designers [121] and independently by Barreto et al. [14]. Later, Cheon et al. proposed a four S-box layers impossible differential [148] and utilized it to attack HC-L1 reduced to six S-box layers. In the related-key setting, Taga et al. utilized a differential characteristic in the key scheduling of HC-L1 to attack eight S-box layers [136].

In this chapter, we assess the security of HC-L1 against Meet-in-the-Middle Attacks. First, we construct a five S-box layers truncated differential characteristic for HC-L1. Then, we utilize this characteristic as a distinguisher to launch a MitM attack based on the differential enumeration technique against HC-L1 reduced to eight S-box layers. Unlike the majority of existing MitM attack results, the matching step in our attack is performed around the linear transformation. Particularly, in the offline phase, we compute two specific bytes of the input of the linear transformation and store their XOR in a precomputation table. Then, in the online phase, we compute two particular bytes of the output of that linear transformation, compute their XOR, which is equivalent to the XOR of the two input bytes, and look for a match in the precomputation table. If no match is found, the key is discarded. Our attack recovers the master key with data complexity of 2^{49} chosen plaintexts, time complexity of $2^{114.8}$ eight S-box layers HC-L1 encryptions and memory complexity of 2^{106} 64-bit blocks.

The rest of this chapter is organized as follows. Section 4.2 provides a description of HC-L1 and the notation adopted in this chapter. Section 4.3 describes our five S-box layers distinguisher and how it is used to launch our MitM attack to recover the master key. Then, the chapter is summarized in Section 4.4.

4.2 Specification of Hierocrypt-L1

HC-L1 is an iterated block cipher that operates on 64-bit blocks and uses a 128-bit key. It adopts a nested SPN construction, which embeds a lower level SPN structure within a higher SPN one. It has six rounds where the last round is slightly different than the others. As shown in Figure 4.1, the higher SPN level of HC-L1 consists of the following three operations:

- AK: Mixes a 64-bit layer key with the 64-bit internal state.
- XS: Two 32×32-bit keyed substitution boxes that are applied simultaneously to the internal state.
- MS: A diffusion layer consisting of a byte-wise linear transformation defined by the following matrix:

$$MDS_H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

The lower SPN level, i.e., the two 32×32-bit XS-boxes, as shown in Figure 4.1, comprises of:

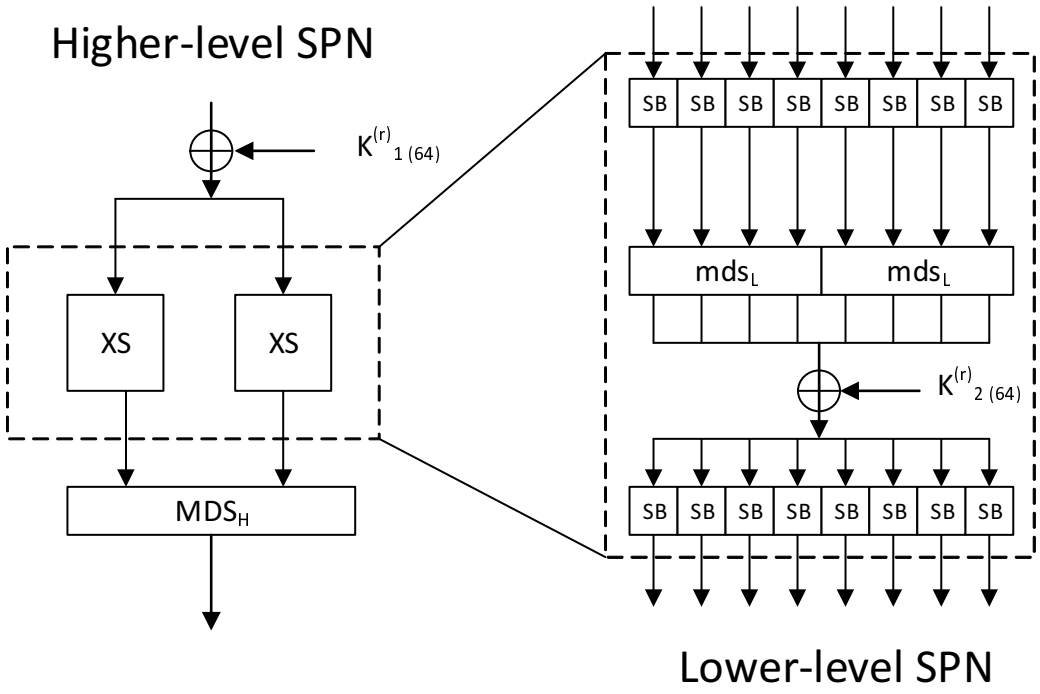

Figure 4.1: One round of Hierocrypt-L1

- SB: A nonlinear byte bijective mapping layer which applies the same 8-bit S-box eight times in parallel.

- MC: A diffusion layer consisting of a byte-wise linear transformation defined by a 4×4 Maximum Distance Separable (MDS) [106, 126] matrix called mds_l .

- AK: The 64-bit layer key is divided into halves and each half is mixed with the 32-bit internal state of one XS box.

- SB: Another nonlinear byte bijective mapping layer which applies the same 8-bit S-box eight times in parallel.

Each round of HC-L1 includes two S-box layers. The last round of HC-L1 is an output transformation where the MS linear operation is substituted by an XOR with a layer key. The full encryption function of HC-L1 where the ciphertext C is computed from the plaintext P is

given by:

$$C = AK[K_1^{(7)}] \circ ((SB \circ AK[K_2^{(6)}] \circ MC \circ SB) \circ AK[K_1^{(6)}]) \\ \circ \cdots \circ (MS \circ (SB \circ AK[K_2^{(1)}] \circ MC \circ SB) \circ AK[K_1^{(1)}])(P)$$

To facilitate the understanding of our attacks, we represent the internal state of HC-L1 as a 4×2 matrix, as depicted in Figure 4.2, where each 8-bit word in the i^{th} row and the j^{th} column of this matrix represents a state byte. Consequently, MC, similar to the MixColumns operation in AES, operates column-wise and MS affects the entire matrix. Moreover, we exploit the fact that both the linear transformations (MC, MS), and the key addition AK are linear and swap their order. In such case, the input data is first XORed with an equivalent layer key, denoted by EK_i, and then the linear transformation is applied. The equivalent layer key at any given S-box layer i is evaluated by $EK_i = MC^{-1}(K_i)$ when i is odd and $EK_i = MS^{-1}(K_i)$ when i is even. In addition, we use the following property of the S-box:

Proposition 4.1 (Differential Property of a bijective S-Box "S") *Given Δ_i and Δ_o two non-zero differences in $\mathbb{F}_{256}$, the equation: $S(x) + S(x + \Delta_i) = \Delta_o$ has one solution on average. This property also applies to S^{-1}.*

Key schedule. The input to the key schedule is the 128-bit master key and the output is 13 64-bit layer keys (1 key per S-box layer in addition to the final key). The master key initializes the first key state denoted by $V_{1(32)}^{(-1)} \| V_{2(32)}^{(-1)} \| V_{3(32)}^{(-1)} \| V_{4(32)}^{(-1)}$ which then undergoes eight rounds relying on a Feistel construction and linear transformations to generate the layer keys where the first round is a bit special as it omits a linear function and does not produce any layer keys. Then, depending on the employed function, the other rounds form two groups, which we mark as 'type A' and 'type B'. The two 32-bit key state words $V_{3(32)}$ and $V_{4(32)}$ are updated linearly in each round, while the other two 32-bit key state words $V_{1(32)}$ and $V_{2(32)}$ are updated using a Feistel construction with additional input from $V_{3(32)}$ and $V_{4(32)}$. Specifically, as shown in Figure 4.3,

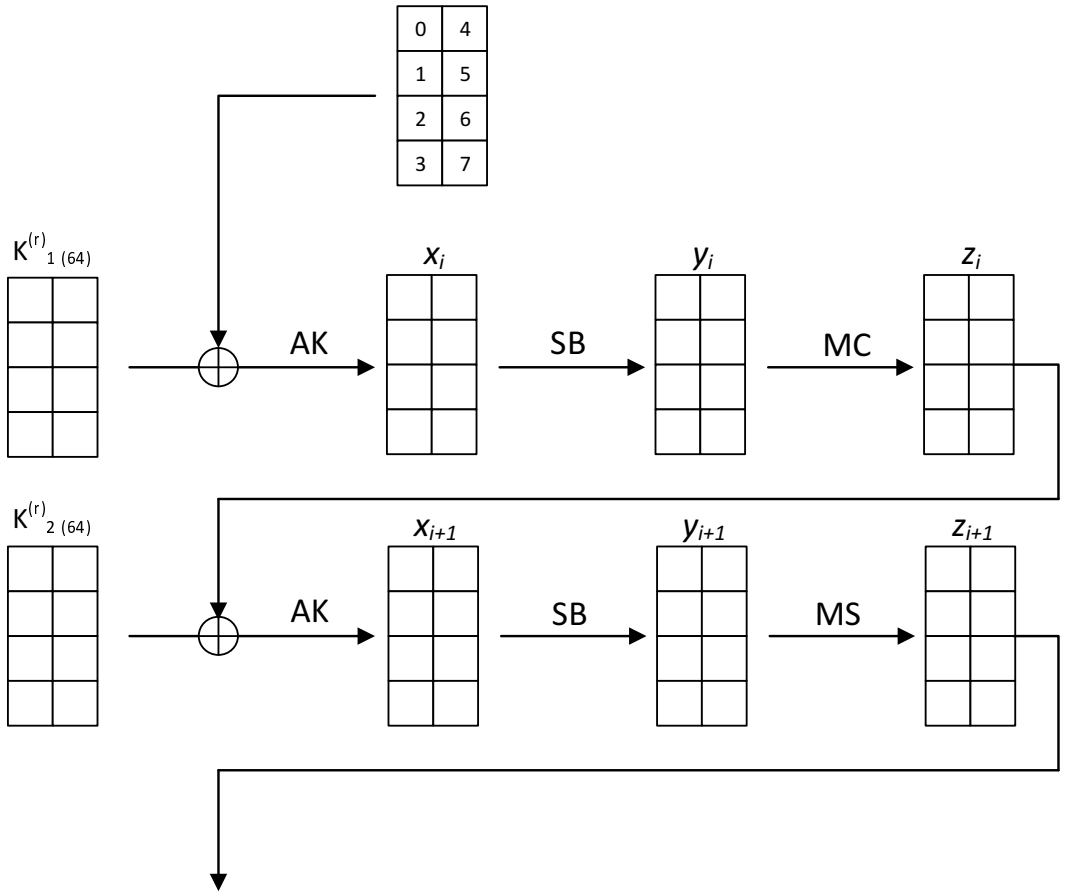

Figure 4.2: Alternative representation of one round of Hierocrypt-L1

one round of 'type A' of the key schedule can be described by:

$$
\begin{aligned}
(V_{3(32)}^{(r)}, V_{4(32)}^{(r)}) &\leftarrow L(V_{3(32)}^{(r-1)}, V_{4(32)}^{(r-1)});\\
V_{1(32)}^{(r)} &\leftarrow V_{2(32)})^{(r-1)};\\
V_{2(32)}^{(r)} &\leftarrow V_{1(32)}^{(r-1)}; \oplus F_{\sigma}(V_{2(32)}^{(r-1)} \oplus V_{3(32)}^{(r)}), \quad r = 0, 1, \cdots, 7
\end{aligned}
$$

where L is a linear function and the function F_{σ} is a level of S-boxes succeeded by another linear transformation. Then, the 64-bit layer keys $K_{1(64)}^{(r)}$ $(k_{1(32)}^{(r)} \| k_{2(32)}^{(r)})$ and $K_{2(64)}^{(r)}$ $(k_{3(32)}^{(r)} \| k_{4(32)}^{(r)})$ are

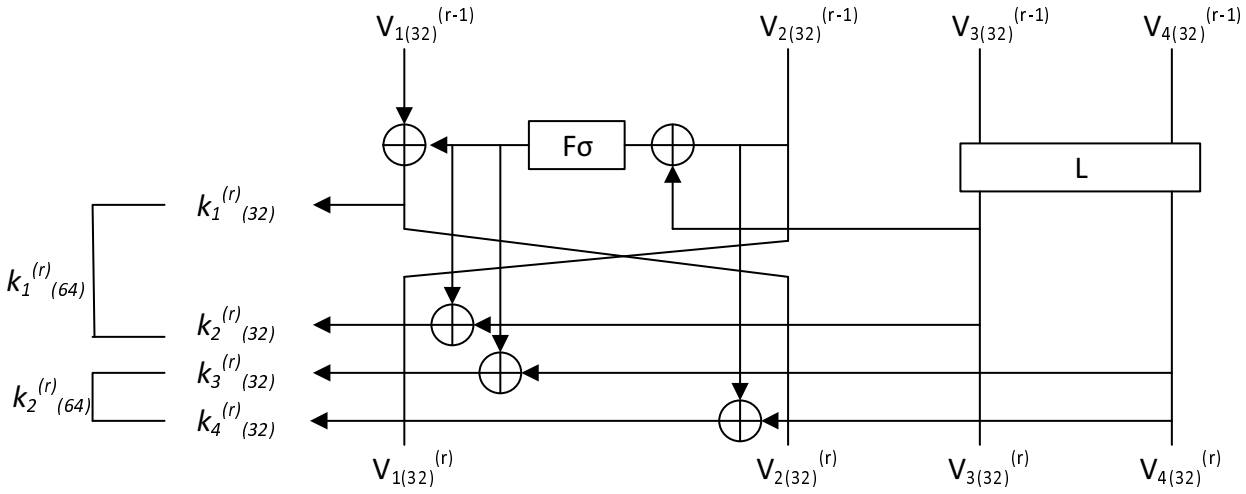

Figure 4.3: One Round of Hierocrypt-L1 key schedule

generated in every round of 'type A' as follows:

$$k^{(r)}_{1(32)} \leftarrow V^{(r-1)}_{1(32)} \oplus F_\sigma(V^{(r-1)}_{2(32)} \oplus V^{(r)}_{3(32)}); \tag{4.1}$$

$$k^{(r)}_{2(32)} \leftarrow V^{(r)}_{3(32)} \oplus F_\sigma(V^{(r-1)}_{2(32)} \oplus V^{(r)}_{3(32)}); \tag{4.2}$$

$$k^{(r)}_{3(32)} \leftarrow V^{(r)}_{4(32)} \oplus F_\sigma(V^{(r-1)}_{2(32)} \oplus V^{(r)}_{3(32)}); \tag{4.3}$$

$$k^{(r)}_{4(32)} \leftarrow V^{(r)}_{4(32)} \oplus V^{(r-1)}_{2(32)}, \qquad r = 0, 1, \cdots, 7 \tag{4.4}$$

The round function of 'type B' is almost equivalent to the inversion of the 'type A' round function but the linear function that operates on $V_{3(32)}$ and $V_{4(32)}$ is different. It is to be noted that in our attacks, we number the layer keys sequentially from K_1 up to K_{13} where $K_i = K^{\lceil i/2 \rceil}_{1(64)}$ when i is odd and $K_i = K^{\lceil i/2 \rceil}_{2(64)}$ when i is even.

For further details regarding the S-box, the linear transformations or the key schedule, the reader is referred to [148].

The following notations are used throughout this chapter:

- x_i: The internal state at the input of layer i

- y_i: The internal state after the SB of layer i.
- z_i: The internal state after the MC (resp. MS) of layer i when i is odd (resp. even).
- $z_i^{'}$: The internal state after the AK of layer i with an equivalent key EK_i.
- $x_i[j]$: The j^{th} byte of the state x_i, where $j = 0, 1, \cdots, 7$, and the bytes are indexed as described in Figure 4.2.
- $x_i[j \cdots k]$: The bytes between the j^{th} position and k^{th} position of the state x_i.
- Δx_i,$\Delta x_i[j]$: The difference at state x_i and byte $x_i[j]$, respectively.
- $X_{(n)}$: An n-bit word X. Such notation is specifically used in describing the key schedule.

The memory and time complexities of our attack are measured as 64-bit HC-L1 blocks and round-reduced HC-L1 encryptions, respectively.

4.3 A Differential Enumeration MitM Attack on HC-L1

Generally, in a MitM attack, a round reduced block cipher E_K is split into three successive parts, such that $E_K = E^2_{K_2} \circ E_m \circ E^1_{K_1}$, where E_m exhibits a distinguishing property. The exploited property is used to identify the correct key by checking whether each guess of subkey (K_1, K_2) yields the property or not. In our attacks, we use a truncated differential characteristic as the distinguishing property, such that its input is a δ-set [50] captured by Definition 4.1. While in most of the published MitM attacks the matching is performed around a specific byte or word, adopting such approach on HC-L1 requires a time complexity that exceeds the key exhaustive search. Therefore, as explained in details below, we opt for matching on a single equation that relates two input bytes of the linear transformation MS with two bytes at its output.

Definition 4.1 (δ-set of HC-L1) *Let a δ-set be a set of 256 HC-L1 states that are all different in one state byte (the active byte) and all equal in the other state bytes (the inactive bytes).*

In our MitM attack, we use the five S-box layers distinguisher embedded in the truncated differential characteristic, illustrated in Figure 4.4. It starts at x_2 and ends at the input of the linear transformation MS of layer 6, i.e., $z_6^{'}$. We exploit the simplicity of the MS operation by observing the below equations of two of its output bytes:

$$z_i[0] = y_i[0] \oplus y_i[2] \oplus y_i[4] \oplus y_i[5] \oplus y_i[6] \tag{4.5}$$

$$z_i[7] = y_i[0] \oplus y_i[2] \oplus y_i[4] \oplus y_i[6] \oplus y_i[7] \tag{4.6}$$

Therefore, from (4.5) and (4.6), we have:

$$z_i[0] \oplus z_i[7] = y_i[5] \oplus y_i[7] \tag{4.7}$$

Consequently, it follows that $x_7[0] \oplus x_7[7] = z_6'[5] \oplus z_6'[7]$ (see Figure 4.4) which is the single equation upon which the matching is performed as will be explained in the attack procedure. Proposition 4.2, below, is the core of our attack.

Proposition 4.2 *If a message m belongs to a pair of states conforming to the truncated differential characteristic of Figure 4.4, then the ordered sequence of differences $\Delta z_6^{'}[5] \oplus \Delta z_6^{'}[7]$ obtained from the δ-set constructed from m in $x_2[3]$ is fully determined by the following 14 bytes: $x_2[3], \Delta x_2[3], x_3[1,3,4,6], y_5[4 \cdots 7], \Delta y_6[5], \Delta y_6[7], y_6[5]$ and $y_6[7]$.*

Proof. The proof is based on rebound-like arguments adopted from the cryptanalysis of hash functions [111] and used in [58]. Assuming that $(m,\ m^{'})$ is a pair that follows the truncated differential characteristic in Figure 4.4. In the sequel, we manifest that knowing these specific 14 bytes is sufficient to compute the ordered sequence of differences $\Delta z_6^{'}[5] \oplus \Delta z_6^{'}[7]$. To conform to the differential characteristic in Figure 4.4, the 14 bytes $x_2[3], \Delta x_2[3], x_3[1,3,4,6], y_5[4 \cdots 7], \Delta y_6[5], \Delta y_6[7]$, $y_6[5]$ and $y_6[7]$ can take $2^{8\times 13} = 2^{104}$ possible values only. This is because $\Delta y_6[5]$ must equal $\Delta y_6[7]$ in order to result in a difference in just $x_7[0,7]$. Then for each of these 2^{104} values, we can determine all the differences shown in Figure 4.4.

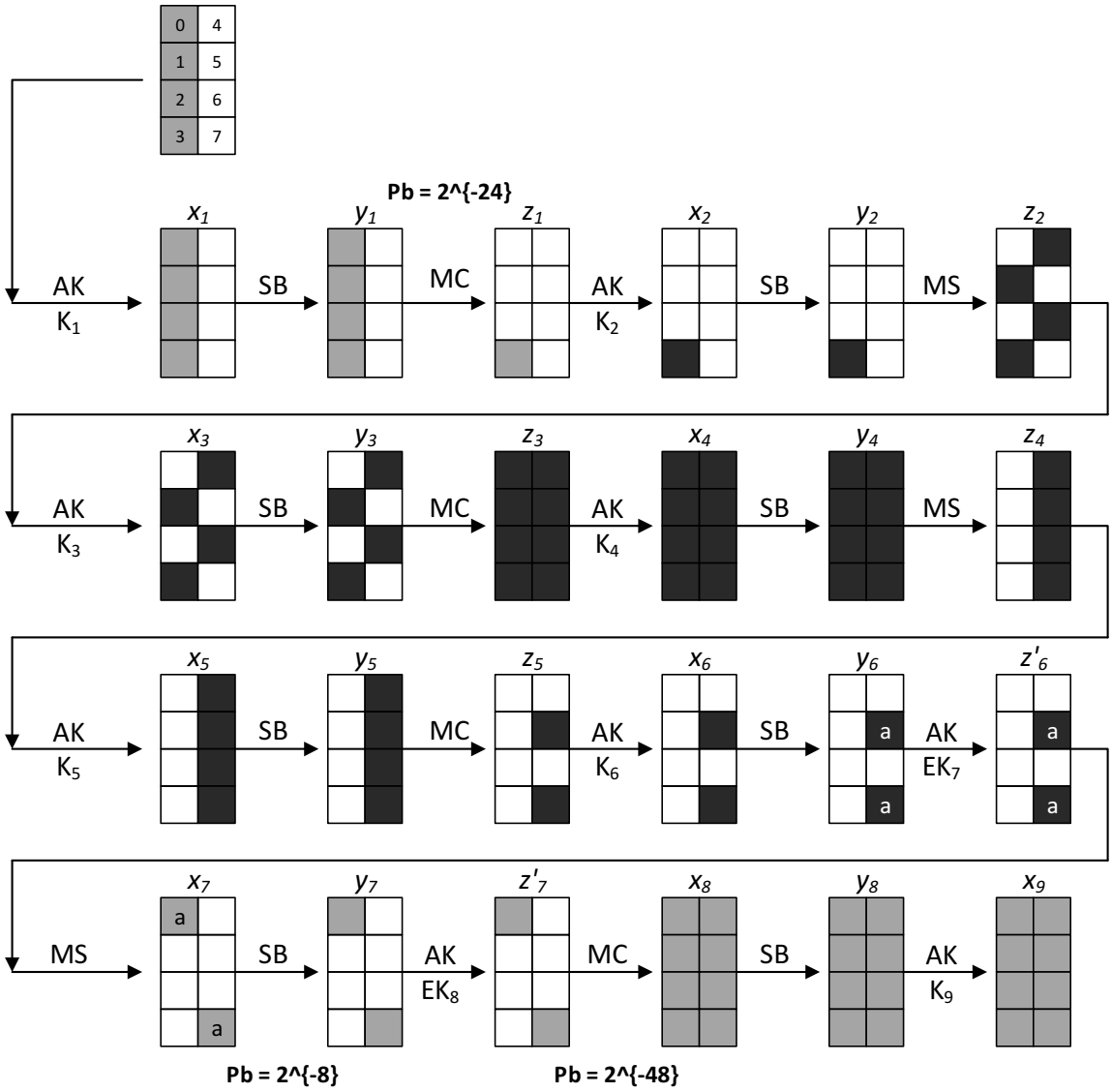

Figure 4.4: The differential characteristic used in the MitM attack on HC-L1 using differential enumeration. Offline bytes are colored in black while online bytes are colored in gray.

- The value $x_2[3]$ with the difference $\Delta x_2[3]$ enable us to bypass the S-box of layer 2 and then propagate the difference linearly through MS to compute $\Delta x_3[1, 3, 4, 6]$.

- By knowing $x_3[1, 3, 4, 6]$, we can bypass the S-box of layer 3 to reach y_3, then linearly through MC to compute $\Delta x_4[0 \cdots 7]$.

- Similarly in the other direction, the differences $\Delta y_6[5]$ and $\Delta y_6[7]$ with the values $y_6[5]$ and $y_6[7]$ enable us to bypass the S-box of layer 6 and compute the difference $\Delta z_5[5, 7]$,

then linearly through MC^{-1} we compute $\Delta y_5[4 \cdots 7]$.

- By knowing $y_5[4 \cdots 7]$, we bypass the S-box of layer five to compute $\Delta z_4[4 \cdots 7]$ and then linearly through MS^{-1} we can compute $\Delta y_4[0 \cdots 7]$.

- Now, we have the differences $\Delta x_4[0 \cdots 7]$ and $\Delta y_4[0 \cdots 7]$. Hence, by the differential property of the HC-L1 S-box (Proposition 4.1), there is, on average, one solution for each of the eight bytes of x_4.

To build the ordered sequence for each of the 2^{104} possible values of the 14 bytes from proposition 4.1, we consider all the $2^8 - 1$ possible values for the difference $\Delta x_2[3]$ and propagate them until $z_6^{'}$ with the help of the internal state solutions we have. This creates an ordered sequence of $2^8 - 1$ differences in $\Delta z_6^{'}[5, 7]$.

Attack Procedure. Similar to other MitM attacks, our attack has 2 phases; an offline phase and an online phase that result in recovering the 64-bit last layer key K_9, 2 bytes of $EK_8 = MC^{-1}(K_8)$ and 4 bytes of K_1.

Offline Phase. In the offline phase, we compute all the 2^{104} values of $\Delta z_6^{'}[5] \oplus \Delta z_6^{'}[7]$ determined by the 14 bytes listed in proposition 4.1 and store them in a precomputation table T.

Online Phase. The online phase can be divided into two stages; data collection and key recovery. The data collection stage aims at finding pairs of messages that follow the truncated differential characteristic in Figure 4.4. Then, for each of the found pairs, a δ-set is created, its corresponding ordered sequence is computed and tested for a match in T to identify the correct key in the key recovery stage.

Data collection. To generate one pair of messages that conforms to the eight S-box layers truncated differential characteristic in Figure 4.4, the encryption oracle is queried with struc-

tures of chosen plaintexts. In a given structure, bytes $[0 \cdots 3]$ take all the 2^{32} possible values while the other four bytes are fixed to some, possibly different, constants and hence, each structure generates $2^{32} \times (2^{32} - 1)/2 \approx 2^{63}$ pairs. Our eight S-box layers truncated differential characteristic has a probability of $2^{-3\times3\times8-1\times8} = 2^{-80}$ because of the three $4 \rightarrow 1$ transitions over MC in addition to the probability that $\Delta x_7[0]$ equals $\Delta x_7[7]$, marked as a in Figure 4.4. Consequently, in order to find one pair that follows our chosen 2^{-80} probability truncated differential characteristic, 2^{80} pairs are needed which is equivalent to 2^{80-63}, i.e., 2^{17} structures of 2^{32} messages, each. Briefly, $2^{17+32} = 2^{49}$ messages are sent to the encryption oracle to generate the required 2^{80} pairs. It is to be noted that the distinguisher was chosen to start at $x_2[3]$ because this specific byte results into just four differences, i.e., $z_2[1, 3, 4, 6]$ after the application of the MS transformation (cf. 4^{th} column of MDS_H).

Key Recovery. The key bytes: $K_9[0 \cdots 7]$, $EK_8[0, 7]$ and $K_1[0 \cdots 3]$ can take $2^{(2+1+1)\times8} = 2^{32}$ possible values for each of the 2^{80} pairs. This is justified as follows:

- The difference $\Delta y_7[0, 7]$ is guessed and propagated linearly through MC to compute $\Delta x_8[0 \cdots 7]$.
- The difference $\Delta y_8[0 \cdots 7]$ is equal to the difference $\Delta x_9[0 \cdots 7]$ which, in turn, is the difference in the ciphertext pair.
- As we have the difference $\Delta x_8[0 \cdots 7]$ and $\Delta y_8[0 \cdots 7]$, the differential property of the S-box is used to deduce a solution for each byte of x_8 and y_8 which yields 2^{16} key candidates for the whole K_9 by simply XORing y_8 with the ciphertext.
- Then, $x_8[0 \cdots 7]$ is propagated linearly through MC^{-1} to deduce $z'_7[0 \cdots 7]$.
- Afterwards, the difference $\Delta y_6[5, 7]$ which, as explained before, assumes 2^8 values only is guessed and propagated through MS to get the difference $\Delta x_7[0, 7]$.
- As we have $\Delta x_7[0, 7]$ and $\Delta y_7[0, 7]$ were already guessed in the first step above, the

differential property of the S-box is used to deduce a solution for $y_7[0,7]$ which with $z_7'[0 \cdots 7]$, computed above, enables us to deduce 2^8 candidates for $EK_8[0,7]$.

- Next, the difference $\Delta x_2[3]$ is guessed and propagated linearly through MC^{-1} to compute $\Delta y_1[0 \cdots 3]$.
- The difference $\Delta x_1[0 \cdots 3]$ is actually the difference in the plaintext pair. Therefore, knowing the differences $\Delta x_1[0 \cdots 3]$ and $\Delta y_1[0 \cdots 3]$ enables us to deduce 2^8 candidates for $K_1[0 \cdots 3]$ using the differential property of the S-box.

Overall, guessing four bytes helps deduce 14 key bytes. In other words, these 14 key bytes have just 2^{32} possible values for each of the 2^{80} pairs. Then, in order to recover the key, we enumerate each of the 2^{80} candidate pairs we obtained in the data collection stage and deduce the corresponding 2^{32} possible key suggestions. Next, we build the plaintext δ-set and compute its corresponding ordered sequence of $\Delta x_7[0] \oplus \Delta x_7[7]$ and look for a match in T and if no match is found, this key suggestion is discarded.

A valid ordered sequence can be generated by a wrong key with a negligible probability, $2^{80+32+104-255\times 8} = 2^{-1824}$, which can be relaxed. Therefore, we use the partial sequence matching idea proposed in [11]. Instead of matching $2^8 - 1$ bytes ordered sequence, we match b bytes such that $b < 2^8$ and the probability of error, chosen to be 2^{-32}, is still small enough to be able to identity the right key. In that case, the number of required bytes b is calculated by $2^{-32} = 2^{80+32+104-8b}$ yielding $b = 31$. Therefore, it is enough to match 31 bytes of the ordered sequence to identify the right key with a negligible error probability of 2^{-32}.

So far, we have recovered 14 key bytes; the eight bytes of K_9, two bytes of EK_8 and four bytes of K_1. To recover the master key, 6 bytes of EK_8 are guessed to get 2^{48} suggestions for K_8, which, using the key schedule notation, means that there are 2^{48} candidates for the keys

$k^{(4)}_{3(32)}, k^{(4)}_{4(32)}$. Along with K_9 or rather $k^{(5)}_{1(32)}$ and $k^{(5)}_{2(32)}$, these keys are computed as follows:

$$k^{(4)}_{3(32)} = V^{(4)}_{4(32)} \oplus F_\sigma(V^{(3)}_{2(32)} \oplus V^{(4)}_{3(32)}) \tag{4.8}$$

$$k^{(4)}_{4(32)} = V^{(4)}_{4(32)} \oplus V^{(3)}_{2(32)} \tag{4.9}$$

$$k^{(5)}_{1(32)} = V^{(4)}_{1(32)} \oplus F_\sigma(V^{(4)}_{2(32)} \oplus V^{(5)}_{3(32)}) \tag{4.10}$$

$$k^{(5)}_{2(32)} = V^{(5)}_{3(32)} \oplus F_\sigma(V^{(4)}_{2(32)} \oplus V^{(5)}_{3(32)}) \tag{4.11}$$

Then, the 32-bit $V^{(3)}_{2(32)}$ is guessed and $V^{(4)}_{4(32)}$ is computed from equation (5) and, in turn, $V^{(4)}_{3(32)}$ is deduced from equation (4). According to the key schedule, the knowledge of $V^{(4)}_{3(32)}$ and $V^{(4)}_{4(32)}$ results in knowing $V^{(5)}_{3(32)}$ and $V^{(5)}_{4(32)}$. Next, since $V^{(4)}_{1(32)} = V^{(3)}_{2(32)}$, $V^{(4)}_{2(32)}$ is computed from equation (6) and then equation (7) is used as a 2^{-32} filter to get one solution for $V^{(3)}_{2(32)}, V^{(4)}_{1(32)}, V^{(4)}_{2(32)}, V^{(4)}_{3(32)}, V^{(4)}_{4(32)}, V^{(5)}_{3(32)}$ and $V^{(5)}_{4(32)}$. As we recover one full intermediate state of the key schedule and its round is bijective, we can recover the master key and get 2^{48} candidates for the master key corresponding to the 2^{48} K_8 suggestions. The correct master key is found by exhaustively searching through these 2^{48} candidates using two plaintext/ciphertext pairs with no significant impact on the overall time complexity of the attack.

Attack Complexity. The size of the precomputation table T created in the offline phase determines the memory requirement of the attack. T contains 2^{104} ordered sequences, each of $8 \times 31 = 248$ bits by using the partial sequence matching technique. Therefore, the memory complexity of the attack is $2^{104} \times 248/64 \approx 2^{106}$ 64-bit blocks. The data collection stage of the online phase sets the data complexity of the attack to 2^{49} chosen plaintexts. The offline phase time complexity to build T is attributed to executing 2^{104} partial encryptions on 32 messages, which is equivalent to $2^{104+5} \times 11/(8 \times 8) = 2^{106.46}$ encryptions. The online phase time complexity to recover 14 key bytes is determined by the time needed to partially decrypt the 2^5 values in a δ-set with all the 2^{32} key suggestions for all the 2^{80} generated pairs which is equivalent to $2^{80+32+5} \times (4 + 10)/(8 \times 8) = 2^{114.8}$. Finding the correct master key among the 2^{48}

candidates using two plaintext/ciphertext pairs requires $2 \times 2^{48} = 2^{49}$ encryptions. Therefore, the time complexity of the attack is equivalent to $2^{114.8} + 2^{106.46} + 2^{49} \approx 2^{114.8}$.

4.4 Summary

In this chapter, we have analyzed one of the Japanese e-Government 2013 candidate recommended block ciphers; Hierocrypt-L1 using the MitM attack in the single-key setting. Our attack employs the differential enumeration technique and is launched against eight S-box layers using a five S-box layers distinguisher. The attack recovers the master key with data complexity of 2^{49} chosen plaintexts, time complexity of $2^{114.8}$ eight S-box layers Hierocrypt-L1 encryptions and memory complexity of 2^{106} 64-bit blocks. This is the first attack on Hierocrypt-L1 in the single-key setting that reaches eight S-box layers.

Chapter 5

Meet-in-the-Middle Attacks on Reduced-Round Hierocrypt-3

In this chapter, we investigate the resistance of the Hierocrypt-3 block cipher against MitM attacks. Hierocrypt-3 is an SPN block cipher designed by Toshiba Corporation. It operates on 128-bit state using either 128, 192 or 256-bit key. In this chapter, we present two meet-in-the-middle attacks in the single-key setting on the eight S-box layer reduced Hierocrypt-3 with 256-bit key. The first attack is based on the differential enumeration approach where we propose a truncated differential characteristic in the first six S-box layers and match a multiset of state differences at its output. The other attack is based on the original meet-in-the-middle attack strategy proposed by Demirci and Selçuk at FSE 2008 to attack reduced versions of both AES-192 and AES-256. For our attack based on the differential enumeration, the master key is recovered with data complexity of 2^{113} chosen plaintexts, time complexity of 2^{238} 8 S-box layer reduced Hierocrypt-3 encryptions and memory complexity of 2^{218} 128-bit blocks. The data, time and memory complexities of our second attack are 2^{32}, 2^{245} and 2^{239}, respectively. These are the first attacks on eight S-box layer reduced Hierocrypt-3.

Parts of this chapter have been published in [1].

5.1 Introduction

Hierocrypt-3 (HC-3) [47, 120, 148], designed by Toshiba Corporation in 2000, is a 128-bit block cipher that was submitted to the New European Schemes for Signatures, Integrity, and Encryption (NESSIE) project [116]. It is among the Japanese e-Government 2003 recommended ciphers list [46] and the 2013 candidate recommended ciphers list [45]. HC-3 employs a nested SPN structure as proposed in [120]. In a nested SPN structure, each round has two substitution layers with distinct linear transformations and hence is equivalent to two rounds in normal SPN structure.

In the self-evaluation report done by Toshiba Corporation [148], HC-3 was concluded to be sufficiently secure against all well-known attacks at that time. Nevertheless, Barreto et al. presented an improved square attack against reduced round HC-3 [14]. They showed that HC-3 is vulnerable up to six S-box layer for 128-bit key, and up to seven S-box layer for 192, 256-bit keys. These attacks are the best attacks on HC-3 so far. Then, Cheon et al. presented a four S-box layer impossible differential that can be used to attack up to 6 S-box layer of HC-3 [81]. Furuya and Rijmen analyzed the key scheduling of HC-3 and showed many linear relations between the round key bits [128]. Finally, after the introduction of the biclique attack in 2011 [35], Rechberger evaluated the security of HC-3, among other 128-bit block ciphers, against the biclique attacks [125].

In this chapter, we present two Meet-in-the-Middle Attacks on HC-3; the first attack uses the idea of efficient differential enumeration while the second one utilizes the *plain* MitM attack in a data-memory trade-off approach. Contrary to AES, HC-3 alternates between two distinct linear transformations; the first linear transformation, similar to the MixColumns transformation in AES, operates on 4 bytes while the other linear transformation acts on the whole 16-byte state. Nevertheless, we manage to construct a six S-box layer truncated differential characteristic that we use to mount an attack on 8 S-box layer reduced HC-3. In the second attack, we show that if

the input has one active byte in a specific position then after four S-box layers of HC-3, certain bytes of the output can be described by a function of that active byte parameterized by 30 8-bit parameters. We use this 4 S-box layer distinguisher to attack 8 S-box layer reduced HC-3 as well with much less data complexity but higher memory and time complexities.

The rest of this chapter is organized as follows. In Section 5.2, we give the specification of HC-3 and provide the notation used throughout this chapter. Section 5.3 discusses the attack based on the efficient differential enumeration technique where we describe the chosen truncated differential characteristic and the adopted attack procedure. Afterwards in Section 5.4, we provide a brief description of our *plain* MitM attack on HC-3. Finally, we summarize the chapter in Section 5.5.

5.2 Specification of Hierocrypt-3

HC-3 is an iterated nested SPN block cipher that operates on 128-bit states with either 128, 192 or 256-bit key. As described in the previous chapter, in a nested SPN structure, the low level SPN structure is recursively used in the SPN of the higher level. That is, one large S-box is composed of two substitution layers with smaller S-boxes and one linear transformation in the middle of them. Therefore, one round in a nested SPN structure has two substitution layers and hence corresponds to two rounds in a normal SPN structure. The number of rounds in HC-3 varies with the cipher key size. For 128-bit keys, HC-3 has six rounds, for 192-bit keys there are seven rounds, and for 256-bit keys, eight rounds. In all cases, the last round is slightly different from the other rounds.

At the higher SPN level, an HC-3 round, as shown in Figure 5.1, consists of three transformations:

- X: A subkey mixing layer consisting of XORing the 128-bit input with 16-byte subkey.

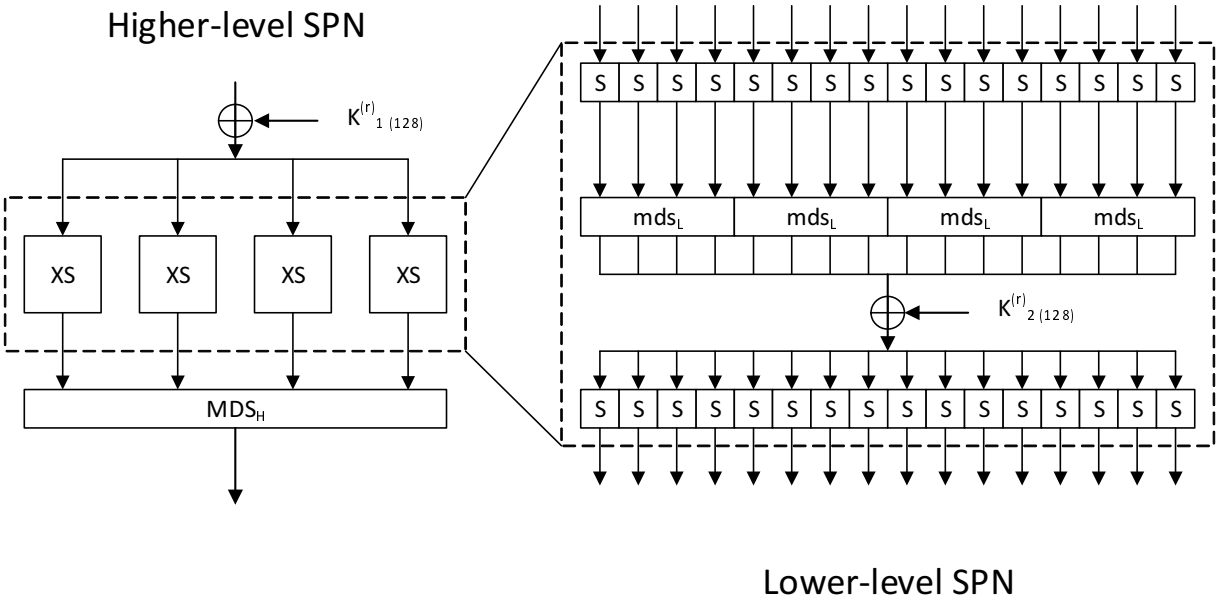

Figure 5.1: One round of Hierocrypt-3.

- XS: A layer of four parallel 32×32-bit keyed substitution boxes.
- H: A linear transformation consisting of XORing 8-bit subdata $x_{i(8)} (\in GF(2)^8; i = 0, 1, \cdots 15)$, which is represented by a matrix MDS_H. The matrix MDS_H and its inverse MDS_H^{-1} are given in Figures 5.2 and 5.3, respectively.

At the lower SPN level, each 32×32-bit XS-box consists of:

- S: A nonlinear layer composed of simultaneous application of four 8-bit S-boxes.
- L: A byte-wise linear transformation defined by a 4×4 matrix called mds_l. L has a branch number of five, i.e., when the input (resp. output) has 1 active byte then the output (resp. input) must have 4 active bytes.
- X: A subkey mixing layer consisting of XORing the 32-bit input with 4-byte subkey.
- S: Another nonlinear layer composed of simultaneous application of four 8-bit S-boxes. Hence, each round consists of two S-box layers and in our attacks below, the counting is done per S-box layer.

$$
\begin{pmatrix} y_{0(8)} \\ y_{1(8)} \\ y_{2(8)} \\ y_{3(8)} \\ y_{4(8)} \\ y_{5(8)} \\ y_{6(8)} \\ y_{7(8)} \\ y_{8(8)} \\ y_{9(8)} \\ y_{10(8)} \\ y_{11(8)} \\ y_{12(8)} \\ y_{13(8)} \\ y_{14(8)} \\ y_{15(8)} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{0(8)} \\ x_{1(8)} \\ x_{2(8)} \\ x_{3(8)} \\ x_{4(8)} \\ x_{5(8)} \\ x_{6(8)} \\ x_{7(8)} \\ x_{8(8)} \\ x_{9(8)} \\ x_{10(8)} \\ x_{11(8)} \\ x_{12(8)} \\ x_{13(8)} \\ x_{14(8)} \\ x_{15(8)} \end{pmatrix}
$$

Figure 5.2: The MDS_H matrix

$$
\begin{pmatrix} x_{0(8)} \\ x_{1(8)} \\ x_{2(8)} \\ x_{3(8)} \\ x_{4(8)} \\ x_{5(8)} \\ x_{6(8)} \\ x_{7(8)} \\ x_{8(8)} \\ x_{9(8)} \\ x_{10(8)} \\ x_{11(8)} \\ x_{12(8)} \\ x_{13(8)} \\ x_{14(8)} \\ x_{15(8)} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} y_{0(8)} \\ y_{1(8)} \\ y_{2(8)} \\ y_{3(8)} \\ y_{4(8)} \\ y_{5(8)} \\ y_{6(8)} \\ y_{7(8)} \\ y_{8(8)} \\ y_{9(8)} \\ y_{10(8)} \\ y_{11(8)} \\ y_{12(8)} \\ y_{13(8)} \\ y_{14(8)} \\ y_{15(8)} \end{pmatrix}
$$

Figure 5.3: The MDS_H^{-1} matrix

In the last round, the H transformation is replaced by a post-whitening key addition. Hence, the full encryption function of the 256-bit key version of HC-3 where the ciphertext C is evaluated from the plaintext P can be described as:

$$C = X[K_1^{(9)}] \circ ((S \circ X[K_2^{(8)}] \circ L \circ S) \circ X[K_1^{(8)}]) \circ \cdots \circ (H \circ (S \circ X[K_2^{(1)}] \circ L \circ S) \circ X[K_1^{(1)}])(P)$$

For the convenience of describing our attacks, we use a different representation of HC-3, similar to the alternative expression used in [81]. As illustrated in Figure 5.4, the state is represented as a 4×4 matrix where each cell in the matrix represents a state byte. In this representation, mds_l operates column-wise, similar to the MixColumns operation in AES, and MDS_H acts on the whole matrix. In some cases, we are also interested in swapping the order of the linear transformations, either H or L, and the XOR with the key X. These operations are linear and hence they can be interchanged, by first XORing the input with an equivalent key and then applying the linear transformation. The equivalent subkey at S-box layer i is denoted by U_i where $U_i = L^{-1}(K_i)$ when i is odd or $U_i = H^{-1}(K_i)$ when i is even. Additionally, we rely on the differential property of a bijective S-box stated in proposition 4.1.

Key schedule. The key state $Z_1 \| Z_2 \| Z_3 \| Z_4$ is 256-bit and undergoes 10 rounds to generate the 17 keys used in the 16 S-box layers in addition to the final key. The first key state is denoted by $Z_1^{(-1)} \| Z_2^{(-1)} \| Z_3^{(-1)} \| Z_4^{(-1)}$ and instantiated with the master key. The first round is special as it omits a linear function and does not generate any round key. The other rounds can be split into two groups which we denote by $G1$ and $G2$. We focus our description on $G1$ as it is used to generate the round keys from S-box layer 1 up to S-box layer 10, which covers our attacked S-box layers. The key state words Z_3 and Z_4 are updated linearly every round, while Z_1 and Z_2 follow a Feistel structure with additional input from Z_3 and Z_4. Particularly, and as illustrated

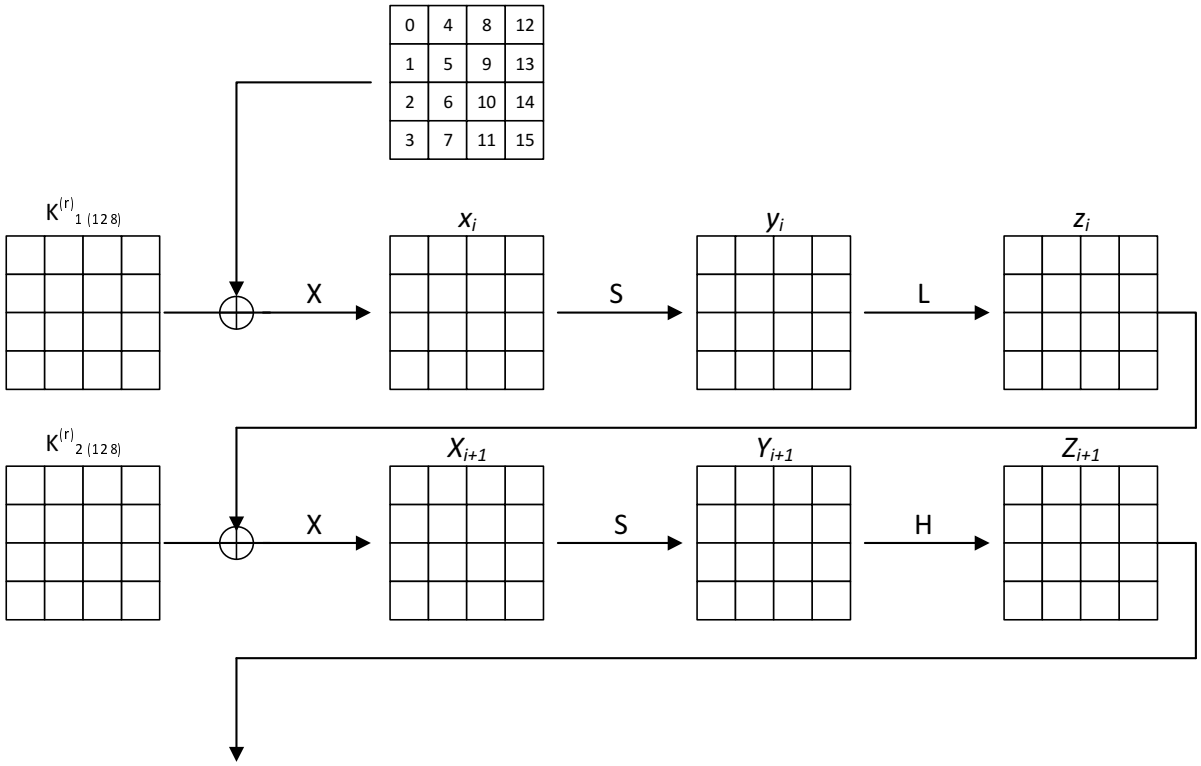

Figure 5.4: Alternative representation of one round of Hierocrypt-3.

in Figure 5.5, the key state words are updated as follows:

$$
\begin{aligned}
(Z_3^{(r)}, Z_4^{(r)}) &= L_1(Z_3^{(r-1)}, Z_4^{(r-1)}) \\
Z_1^{(r)} &= Z_2^{(r-1)} \\
Z_2^{(r)} &= Z_1^{(r-1)} \oplus F_\sigma(Z_2^{(r-1)} \oplus Z_3^{(r)})
\end{aligned}
$$

where L_1 is a specific linear transformation and the function F_σ consists of a level of S-boxes followed by another, different than L_1, linear transformation. Then 64-bit keys $k_1^{(r)}, k_2^{(r)}, k_3^{(r)}$

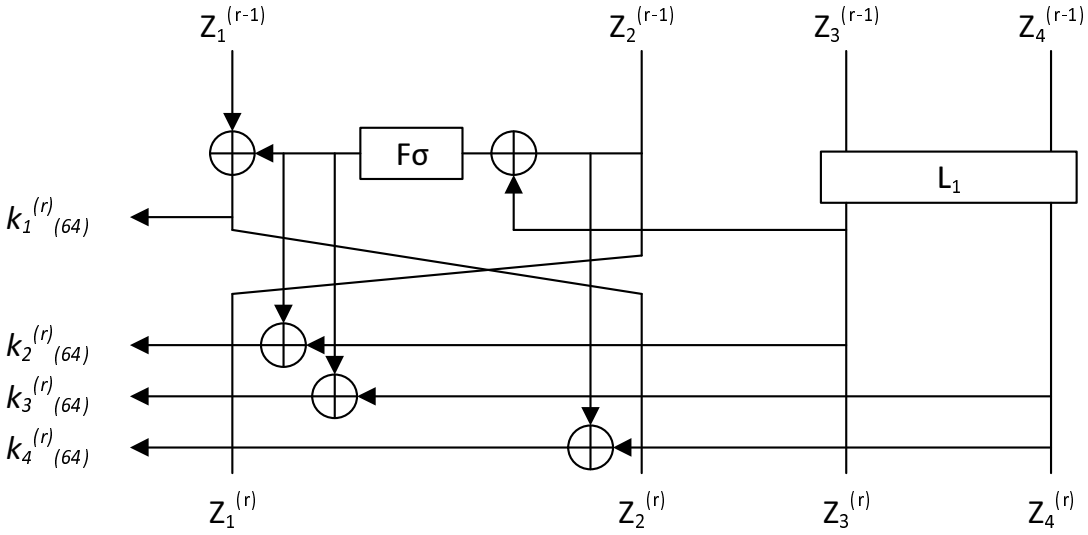

Figure 5.5: 1 Round of Hierocrypt-3 key schedule.

and $k_4^{(r)}$ are generated as follows:

$$
\begin{aligned}
k_1^{(r)} &= Z_1^{(r-1)} \oplus F_\sigma(Z_2^{(r-1)} \oplus Z_3^{(r)}) \\
k_2^{(r)} &= Z_3^{(r)} \oplus F_\sigma(Z_2^{(r-1)} \oplus Z_3^{(r)}) \\
k_3^{(r)} &= Z_4^{(r)} \oplus F_\sigma(Z_2^{(r-1)} \oplus Z_3^{(r)}) \\
k_4^{(r)} &= Z_4^{(r)} \oplus Z_2^{(r-1)}
\end{aligned}
$$

where $r = 1, 2, \cdots, 5$ then the 128-bit $K_{1(128)}^{(r)}$ is set to $k_1^{(r)} \| k_2^{(r)}$ and $K_{2(128)}^{(r)}$ is set to $k_3^{(r)} \| k_4^{(r)}$. It is to be noted that in our attacks, we number the keys sequentially from K_1 up to K_{17} (for the full cipher) where $K_i = K_{1(128)}^{\lceil i/2 \rceil}$ when i is odd and $K_i = K_{2(128)}^{\lceil i/2 \rceil}$ when i is even.

For further details regarding the S-box, the linear transformations or the key schedule, the reader is referred to [148].

5.2.1 Notation

The following notations are used throughout this chapter:

- x_i, y_i: The 16-byte state after the X, S transformations at layer i, respectively.
- z_i: The 16-byte state after the linear transformation at layer i where the linear transformation is L (resp. H) when i is odd (resp. even).
- $x_i[j]$: The j^{th} byte of the state x_i, where $j = 0, 1, \cdots, 15$, and the bytes are indexed as shown in Figure 5.4.
- $x_i[j \cdots k]$: The bytes between the j^{th} position and k^{th} position of the state x_i.
- $\Delta x_i, \Delta x_i[j]$: The difference at state x_i and byte $x_i[j]$, respectively.

We measure memory complexity of our attacks in number of 128-bit HC-3 blocks and time complexity in reduced-round HC-3 encryptions.

5.3 A Differential Enumeration MitM Attack on HC-3

Recall that in a MitM attack, an r-round reduced block cipher is split into 3 consecutive parts of r_1, r_2 and r_3 rounds, $r = r_1 + r_2 + r_3$, such that a particular set of messages may verify a certain property in the middle r_2 rounds. In an offline phase, that particular property is evaluated independently of the keys used in the middle rounds. Then in an online phase, correct key candidates for the r_1 and r_3 rounds are checked whether they verify this distinguishing property or not. In our attacks, the chosen property is a truncated differential characteristic, such that when its input is a δ-set [50] captured by Definition 5.1, the set of each byte of the output states forms an ordered sequence or rather a multiset as in Dunkleman's attack and captured in Definition 5.2.

Definition 5.1 (δ-set of HC-3) *Let a δ-set be a set of 256 HC-3 states that are all different in one state byte (the active byte) and all equal in the other state bytes (the inactive bytes).*

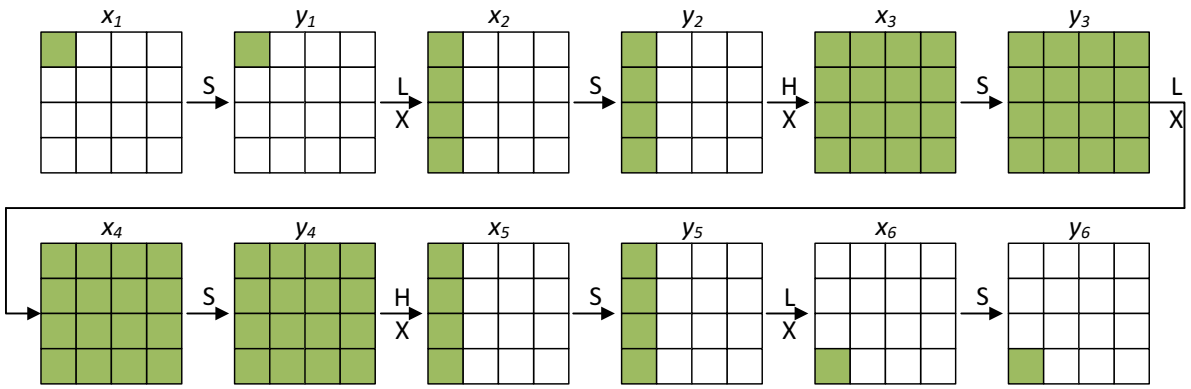

Figure 5.6: The distinguisher used in the MitM attack on HC-3 using differential enumeration.

Definition 5.2 (Multisets of bytes) *A multiset generalizes the set concept by allowing elements to appear more than once. In our case, a multiset of 256 bytes can take* $\binom{2^8 + 2^8 - 1}{2^8} \approx 2^{506.17}$ *different values.*

Our first proposed eight S-box layer MitM attack relies on a 6 S-box layer distinguisher. The distinguisher, as illustrated in Figure 5.6, starts at x_1 and ends at the S-box transformation in layer 6, i.e., y_6. Proposition 5.1, below, is the core of our attack.

Proposition 5.1 *If a message* m *belongs to a pair of states conforming to the truncated differential characteristic of Figure 5.6, then the multiset of differences* $\Delta y_6[3]$ *obtained from the* δ*-set constructed from* m *in* $x_1[0]$ *is fully determined by the following 27 bytes:* $\Delta y_1[0]$, $x_2[0 \cdots 3]$, $x_3[0 \cdots 15]$, $\Delta y_6[3]$, $y_6[3]$ *and* $y_5[0 \cdots 3]$.

Proof. As before, the proof uses rebound-like arguments borrowed from the hash function cryptanalysis [111] and used in [58]. Let (m, m') be a right pair that conforms to the truncated differential characteristic in Figure 5.6. We show in the following how the knowledge of these 27 particular bytes is enough to compute the multisets. The 27 bytes $\Delta y_1[0]$, $x_2[0 \cdots 3]$, $x_3[0 \cdots 15]$, $\Delta y_6[3]$, $y_6[3]$ and $y_5[0 \cdots 3]$ can take $2^{8 \times 27} = 2^{216}$ possible values, and for each of them, we can determine the values of all the differences shown in Figure 5.6. $\Delta y_1[0]$ can be

propagated linearly through L to compute $\Delta x_2[0 \cdots 3]$. With the knowledge of $x_2[0 \cdots 3]$, we can bypass the S-box of layer 2 to reach y_2, then linearly through H to compute $\Delta x_3[0 \cdots 15]$. With the knowledge of $x_3[0 \cdots 15]$, we can bypass the S-box of layer 3 to reach y_3 and then linearly through L to compute $\Delta x_4[0 \cdots 15]$. Similarly in the other direction, $\Delta y_6[3]$ and $y_6[3]$ enable us to bypass the S-box of layer 6 and compute $\Delta z_5[3]$, then linearly through L^{-1} we compute $\Delta y_5[0 \cdots 3]$. With the knowledge of $y_5[0 \cdots 3]$, we bypass the S-box of layer 5 to compute $\Delta z_4[0 \cdots 3]$ and then linearly through H^{-1} we can compute $\Delta y_4[0 \cdots 15]$. By the differential property of the HC-3 S-box (Proposition 4.1), there is, on average, one solution for each of the 16 bytes of the state x_4.

To construct the multiset for each of the 2^{216} possible values of the 27 bytes from proposition 5.1, we consider all the 255 possible values for $\Delta y_1[0]$ and propagate them until y_6 with the help of the internal state solutions we have. This results in a multiset of 255 differences in $\Delta y_6[3]$. As the HC-3 S-box is a permutation over $\mathbb{F}_{256}$, the sequence in $\Delta y_1[0]$ allows to derive the sequence in $\Delta x_1[0]$.

Attack Procedure. Our attack recovers the 128-bit last round key K_9 and 13 bytes of $U_8 = L^{-1}(K_8)$ and is composed of a precomputation phase and an online phase.

Precomputation phase. In the precomputation phase, we iterate over the 2^{216} possible values for the 27 bytes of proposition 5.1 and for each of them, we deduce the possible values of the internal states as discussed in the above proof. These internal state values are then used to generate the multiset. We store all the multisets in a hash table.

Online Phase. The online phase is further divided into two steps; data collection and key recovery. First, we try to find pairs of messages that conform to the truncated differential characteristic in Figure 5.7 in which the previous 6 S-box layer characteristic is placed at the top. Next, the found pairs are used to create a δ-set and test them against the stored hash table to

identify the correct key.

Data collection. In this step, we query the encryption oracle with structures of chosen plaintexts to get enough pairs such that one conforms to the eight S-box layer truncated differential characteristic in Figure 5.7. Each structure is composed of 2^8 plaintexts, where byte 0 takes all the 2^8 possible values while each of the other 15 bytes take any, possibly distinct, fixed value. Thus, each structure is expected to generate $2^8 \times (2^8 - 1)/2 \approx 2^{15}$ pairs. The probability that the truncated differential characteristic is verified is $2^{-7\times8-8\times8} = 2^{-120}$ because of the $16 \rightarrow 9$ transition over L^{-1} ($x_8 \rightarrow z'_7$) and the $9 \rightarrow 1$ transition over H^{-1} ($x_7 \rightarrow z'_6$), see Figure 5.7. While the probability of the former transition over L^{-1} is trivial to deduce, the probability of the latter transition over H^{-1} was deduced by observing that if the input of H^{-1} consists of certain nine active bytes having the same difference, then the output will have just one active byte in a specific position. This position can be either $3, 7, 11$ or 15 depending on the position of the 9 bytes. Hence, the probability of the $9 \rightarrow 1$ transition is equivalent to the probability of 9 active bytes having the same difference, i.e., $2^8/2^{8\times9} = 2^{-8\times8} = 2^{-64}$. This observation is also applicable to H, but the positions of the nine active input bytes and the active output byte differ from the ones corresponding to H^{-1}. Since the probability of the whole truncated differential characteristic is 2^{-120}, then in order to find one pair that conforms to it, we need 2^{120} pairs which means 2^{105} structures of 2^8 messages. Therefore, we ask for the encryption of $2^{105+8} = 2^{113}$ messages to get the required 2^{120} pairs.

Key Recovery. For each of the 2^{120} pairs, we get $2^{8\times(1+9)} = 2^{80}$ suggestions for the 25 key bytes: $K_9[0..15]$ and $U_8[1, 3 \cdots 6, 8, 10, 13, 15]$. This is done as follows: we guess $\Delta y_7[1, 3 \cdots 6, 8, 10, 13, 15]$ and propagate it linearly through L to compute Δx_8. From the other side, Δx_9 is in fact the difference in the ciphertext pair and is also equal to Δy_8. So now, we have Δx_8 and Δy_8, so we use the differential property of the S-box and get a solution for each byte of x_8 and y_8, which enables us to deduce a key candidate for the whole K_9 by XORing the ciphertext with y_8, which in turn, helps compute z'_7. Then, we guess $\Delta y_6[3]$ and propagate it through H to

get $\Delta x_7[1, 3 \cdots 6, 8, 10, 13, 15]$. Once again, we use the differential property of the S-box along with $\Delta y_7[1, 3 \cdots 6, 8, 10, 13, 15]$ which we guessed above to deduce a solution for the 9 bytes $y_7[1, 3 \cdots 6, 8, 10, 13, 15]$ which with $z_7^{'}$ enables us to compute the corresponding 9 bytes in U_8. To summarize this part, we guess 10 bytes that help deduce 25 key bytes.

To compute the multiset at $\Delta y_6[3]$ which is equivalent to $\Delta z_6^{'}[3]$, we noticed that extra key bytes are required. For example, if byte 3 is active in state y_i then, through H, it has an impact on 9 bytes in state z_i, these 9 bytes are $[1, 3 \cdots 6, 8, 10, 13, 15]$ (cf. 4^{th} column in H). However, byte 3, through H^{-1}, is impacted by 11 bytes in state z_i, these 11 bytes are $[0, 2 \cdots 6, 8 \cdots 10, 12, 13]$ (cf. 4^{th} row in H^{-1}). Hence, to compute the multiset, we need to guess the key bytes $[0, 2, 9, 12]$ on top of the above deduced keys. These key bytes are the gray cells in Figure 5.7. This means that for each of the 2^{120} pairs, we have $2^{80+4\times8} = 2^{112}$ key candidates. For each pair and for every key candidate, we build the plaintext δ-set, get the corresponding ciphertexts, compute the multiset and look for a match in the precomputation table. If a match is not found, we can discard that key candidate. The probability of a wrong key producing a valid multiset is given by $2^{120+112+216-467.6} = 2^{-19.6}$ which is negligible. Note that the probability of randomly having a match in the table is $2^{-467.6}$ (and not $2^{-506.7}$) because the number of ordered sequences associated to a multiset is not constant [58].

Now, our attack recovers the 16-byte K_9 and 13 bytes of U_8, which is the equivalent key of K_8 so, in order to recover the master key, we guess 3 bytes of U_8 and get 2^{24} candidates for K_8. This means that we have recovered the 64-bit keys $k_3^{(4)}, k_4^{(4)}, k_1^{(5)}$ and $k_2^{(5)}$. According to the key

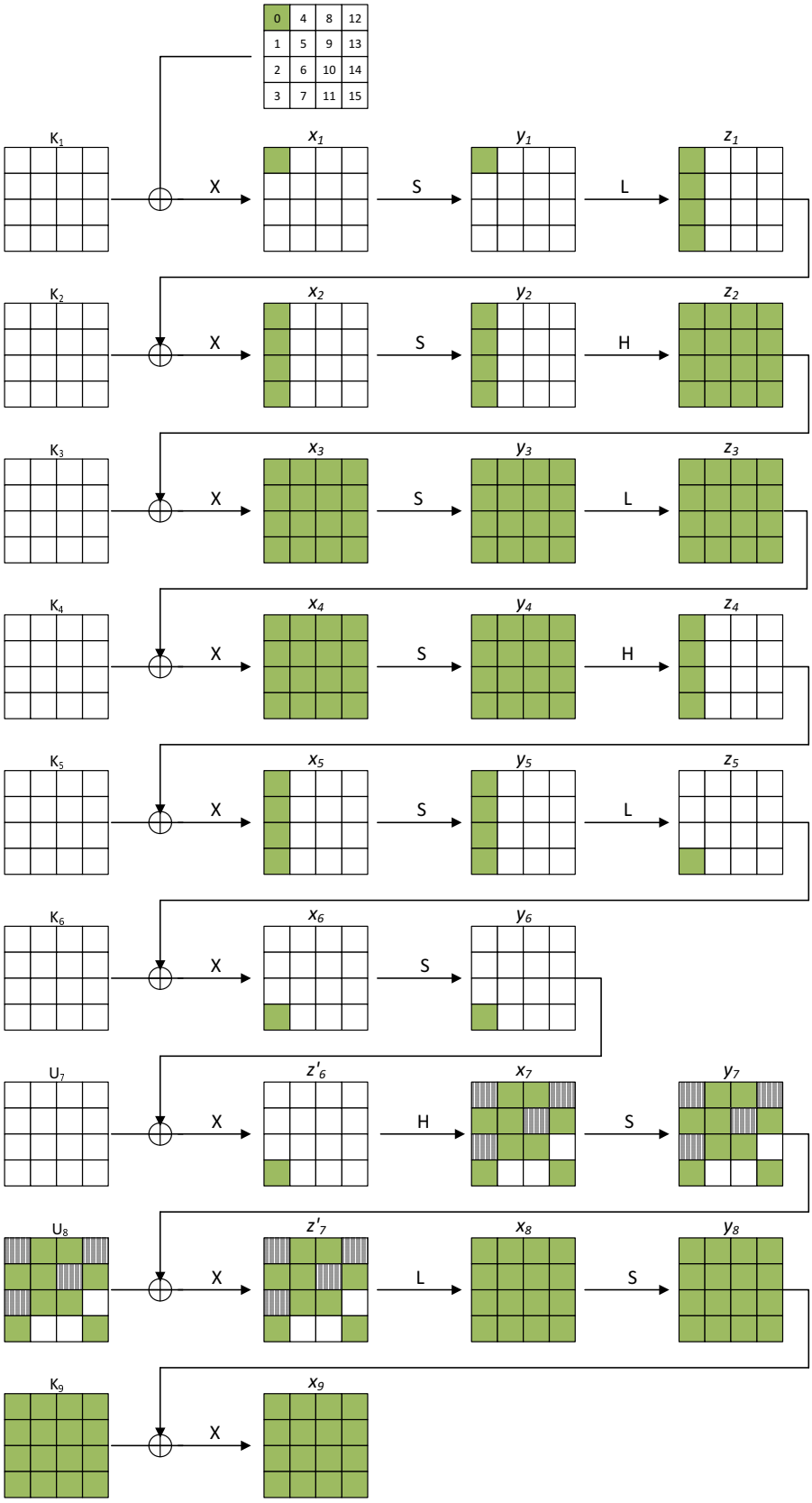

Figure 5.7: Complete eight S-box layer truncated differential characteristic used in the attack using differential enumeration.

schedule, they are calculated as follows:

$$k_3^{(4)} = Z_4^{(4)} \oplus F_\sigma(Z_2^{(3)} \oplus Z_3^{(4)}) \tag{5.1}$$

$$k_4^{(4)} = Z_4^{(4)} \oplus Z_2^{(3)} \tag{5.2}$$

$$k_1^{(5)} = Z_1^{(4)} \oplus F_\sigma(Z_2^{(4)} \oplus Z_3^{(5)}) \tag{5.3}$$

$$k_2^{(5)} = Z_3^{(5)} \oplus F_\sigma(Z_2^{(4)} \oplus Z_3^{(5)}) \tag{5.4}$$

We start by guessing $Z_2^{(3)}$ which is of 64-bit length then from equation (5.2) we compute $Z_4^{(4)}$. From equation (5.1), we compute $Z_3^{(4)}$. From the key schedule, knowing $Z_3^{(4)}$ and $Z_4^{(4)}$ enables us to compute $Z_3^{(5)}$ and $Z_4^{(5)}$. Afterwards from equation (5.3) and considering that $Z_1^{(4)} = Z_2^{(3)}$, we compute $Z_2^{(4)}$. We use equation (5.4) as a 2^{-64} filter and we end up with one solution for $Z_2^{(3)}, Z_1^{(4)}, Z_2^{(4)}, Z_3^{(4)}, Z_4^{(4)}, Z_3^{(5)}$ and $Z_4^{(5)}$. Since we have $Z_1^{(4)}, Z_2^{(4)}, Z_3^{(4)}, Z_4^{(4)}$ we can recover the master key and get 2^{24} candidates for the master key corresponding to the 2^{24} candidates for K_8. We exhaustively search through these master key candidates to find the correct master key with no significant impact on the attack time complexity.

Attack Complexity. The memory requirement of the attack is due to the precomputation table needed to store 2^{216} multisets, each of 512 bits. Hence, the memory complexity of the attack is $2^{216} \times 512/128 = 2^{218}$ 128-bit blocks. The data complexity of the attack is attributed to the data collection step of the online phase where we query the encryption oracle with 2^{113} chosen plaintexts to generate 2^{120} pairs. The time complexity of the offline phase to construct the table is due to performing 2^{216} partial encryptions on 256 messages, which is equivalent to $2^{216+8} \times 2^{-2} = 2^{222}$ encryptions. The time complexity of the online phase to recover 29 key bytes (16-byte K_9 and 13 bytes from U_8) is the time required to partially decrypt the 2^8 values in a δ-set with all the 2^{112} key candidates for all the 2^{120} generated pairs, which is equivalent to $2^{120+112+8} \times 2^{-2} = 2^{238}$. The time complexity of the exhaustive search among the 2^{24} master key candidates using two plaintext/ciphertext pairs is $2 \times 2^{24} = 2^{25}$. Therefore, the

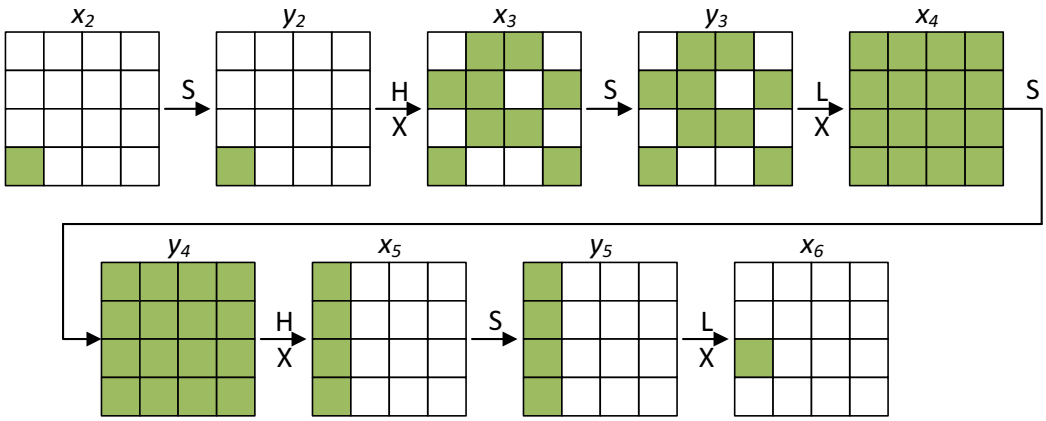

Figure 5.8: The four S-box layer distinguisher used in the MitM attack on HC-3 using Demirci and Selçuk approach.

time complexity of the attack is dominated by the time complexity of the online phase and is equivalent to $2^{238} + 2^{222} + 2^{25} \approx 2^{238}$.

5.4 A *plain* MitM Attack on HC-3

Our second proposed MitM attack on HC-3 is also launched on eight S-box layer of the 256-bit key version in a data-memory trade-off approach. Although its memory and time complexities, as will be shown, are higher than the previous attack by a factor of 2^{21} and 2^7, respectively, it reduces the data complexity dramatically by a factor of 2^{81}. It follows the strategy used by Demirci and Selçuk, which we named above as the *plain* MitM. As illustrated in Figure 5.8, the distinguisher in this attack covers four middle S-box layers starting from x_2 to x_6. Proposition 5.2 is the base of our attack.

Proposition 5.2 *The ordered sequence of differences $\Delta x_6[2]$ obtained from the δ-set constructed by varying $x_2[3]$ is fully determined by the following 30 bytes: $x_2[3]$, $x_3[1, 3 \cdots 6, 8, 10, 13, 15]$, $x_4[0 \cdots 15]$ and $x_5[0 \cdots 3]$.*

Proof. The proof is similar to the proof of proposition 5.1 without using the rebound-like arguments. We show in the following how the knowledge of these 30 particular bytes is enough

to compute the ordered sequence of differences at $x_6[2]$. $\Delta x_2[3]$ and $x_2[3]$ help us bypass the S-box of layer 2 to compute $\Delta y_2[3]$ which is then propagated linearly through H to compute $\Delta x_3[1, 3 \cdots 6, 8, 10, 13, 15]$. With the knowledge of $x_3[1, 3 \cdots 6, 8, 10, 13, 15]$, we can bypass the S-box of layer 3 and then linearly through L, we compute $\Delta x_4[0 \cdots 15]$. With the knowledge of $x_4[0 \cdots 15]$, we can bypass the S-box of layer 4 and once again linearly through H, we compute $\Delta x_5[0 \cdots 15]$. With the knowledge of $x_5[0 \cdots 3]$, we can bypass the S-box of layer 5 in these 4 bytes to reach $y_5[0 \cdots 3]$ and then linearly through L to compute $\Delta x_6[2]$. The byte where the distinguisher starts was chosen such that it minimizes the number of parameters. $x_2[7, 11, 15]$ are other possible positions that would result in the same number of parameters. The byte where the ordered sequence is computed was chosen to minimize the number of key guesses in the online phase. $x_6[6, 10, 14]$ are other positions that would require guessing the same number of key bytes.

Attack Procedure. In this attack, we recover the 16-byte K_9, 9 bytes of $U_8 = L^{-1}(K_8)$, 1 byte of $U_7 = L^{-1}(K_7)$ and 4 bytes of K_1. Similar to the previous attack, this one is also composed of a precomputation phase and an online phase. The precomputation phase in this attack is similar to the precomputation phase in the previous one, we build a hash table for the ordered sequence of $\Delta x_6[2]$ by iterating over all the possible values of the 30 parameters. The online phase is a bit different and thus explained in details below.

Online Phase. In the online phase, we choose an arbitrary plaintext as P^0, guess 4 key bytes $K_1[0 \cdots 3]$ and partially encrypt P^0 through the first layer to reach z_1. We create the δ-set at $z_1[3]$, which is equivalent to creating the δ-set at $x_2[3]$, and decrypt it backward to identify the plaintexts forming the δ-set containing P^0 and ask for their corresponding ciphertexts. From these ciphertexts and by guessing 26 key bytes, we compute the ordered sequence of $\Delta x_6[2]$ and look for a match in the stored hash table. The 26 key bytes to be guessed are $K_9[0 \cdots 15]$, $U_8[0, 1, 3, 6, 7, 10, 11, 13, 15]$ and $U_7[2]$ as depicted in Figure 5.9. The probability for a false positive, i.e. a wrong key guess resulting in a match, is given by $2^{8\times30-2040} = 2^{-1800}$ and as

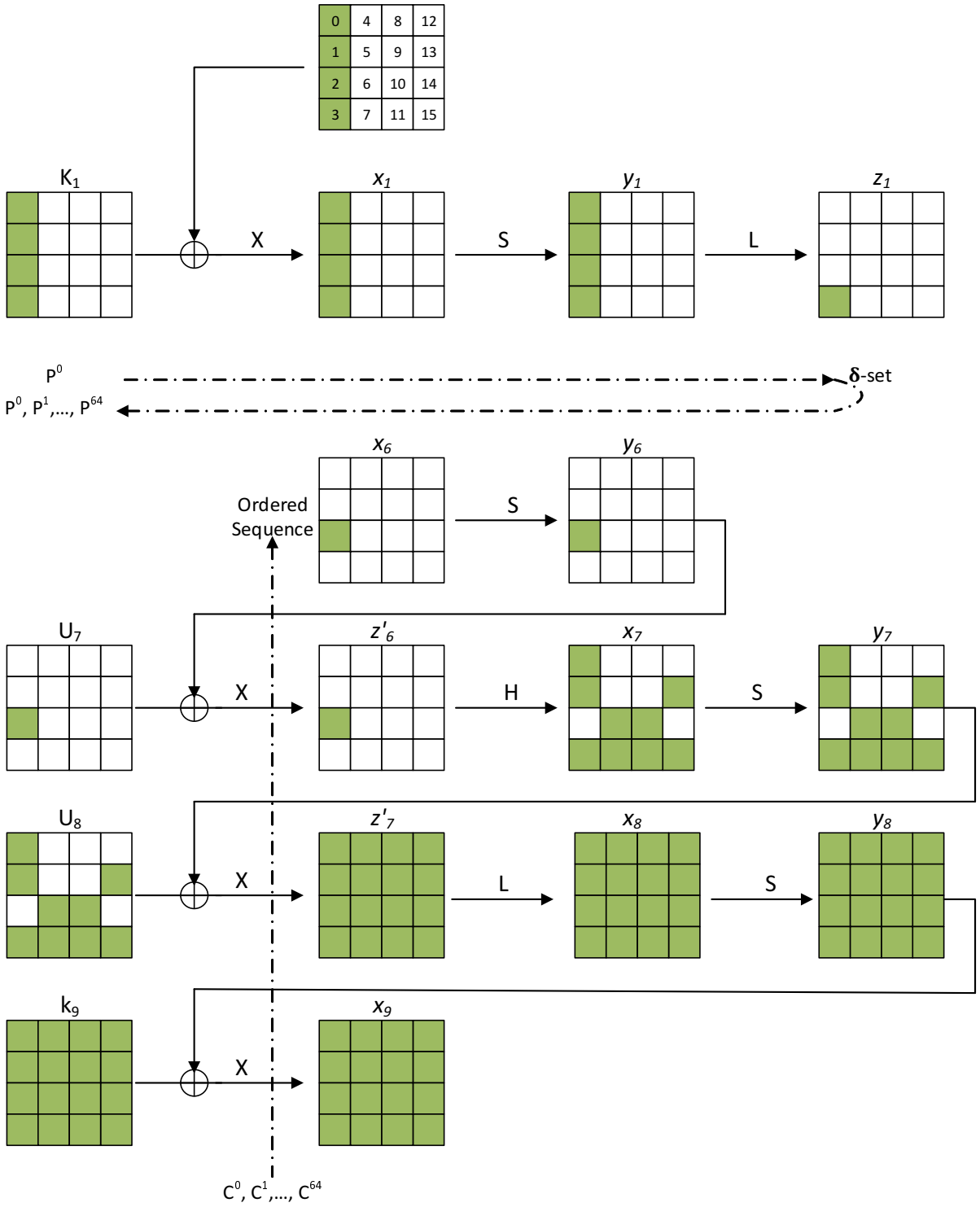

Figure 5.9: The online phase of the eight S-box layer MitM attack on HC-3 using Demirci and Selçuk approach.

we try 2^{240} key candidates, we expect that only $2^{240-1800} = 2^{-1560}$ remain after this step. We notice that the probability of a wrong key producing a valid $2^8 - 1$ bytes ordered sequence is negligible and can be relaxed. Hence, we use the partial sequence matching idea proposed in [11]. Instead of matching $2^8 - 1$ bytes ordered sequence, we match b bytes such that $b < 2^8$ and the probability of error, chosen to be 2^{-32}, is still small enough to be able to identity the right key. In that case, the number of required bytes b is calculated by $2^{-32} = 2^{8\times30+240-8b}$, which yields $b = 64$. This means that it is enough to match 64 bytes of the ordered sequence to identify the right key with a negligible error probability of 2^{-32}. It should be noted that in this attack, we opted for using the ordered sequence rather than the multiset because in the case of multiset, the probability of error is not small enough to identify the right key. Finally, as we recover 9 bytes from U_8, we have 2^{56} candidates for K_8 and following the same approach as in the previous attack, we get 2^{56} candidates for the master key, which we exhaustively search to retrieve the correct master key with no major impact on the overall attack time complexity.

Attack Complexity. The memory requirement of the attack is due to the precomputation table needed to store the partial ordered sequence and is estimated to be $2^{240} \times 64/128 = 2^{239}$ 128-bit blocks. The data complexity of the attack is 2^{32} chosen plaintexts. The time complexity of the offline phase is equivalent to $2^{240} \times 65 \times 2^{-2} \approx 2^{244}$ encryptions. The time complexity of the online phase to recover 30 key bytes (The 16-byte K_9, 4 bytes from K_1, 9 bytes from U_8 and 1 byte from U_7) is equivalent to $2^{240} \times 65 \times 2^{-2} = 2^{244}$. Therefore, the total time complexity of the attack is $2^{244} + 2^{244} + 2^{57} \approx 2^{245}$.

5.5 Summary

In this chapter, we have presented two MitM attacks on the 256-bit key version of one of the Japanese e-Government candidate recommended block ciphers, i.e., Hierocrypt-3. The first attack employs the differential enumeration technique while the second one follows the strategy of Demirci and Selçuk. Both attacks are mounted against eight S-box layers and the complexities

of the first attack are 2^{113} chosen plaintexts, 2^{238} 8 S-box layer reduced Hierocrypt-3 encryptions and 2^{218} 128-bit blocks of memory. The other attack has much less data complexity of 2^{32} chosen plaintexts but higher time and memory complexities of 2^{245} encryptions and 2^{239} 128-bit blocks, respectively. We have noticed that the first attack based on the differential enumeration technique works in the chosen ciphertext context as well with exactly the same data, time and memory complexities. These are the best attacks on the 256-bit version of Hierocrypt-3 in the single-key setting.

Chapter 6

Impossible Differential Attacks on SPARX-64/128

In this chapter, we present an impossible differential attack on SPARX-64/128 which is an ARX-based block cipher with 64-bit block size and 128-bit key. It was published in ASIACRYPT 2016 as one of the instantiations of a family of ARX-based block ciphers with provable security against single-characteristic differential and linear cryptanalysis. In particular, we present 12 and 13-round impossible distinguishers on SPARX-64/128 that can be used to attack 15 and 16-round SPARX-64/128 with post-whitening keys, respectively. While the 15-round attack starts from round zero, the 16-round one, exploiting the key schedule, has to start from round two.

Parts of this chapter have been published in [5].

6.1 Introduction

SPARX is a family of ARX-based block ciphers that was published in ASIACRYPT 2016 [62]. It was designed with the goal of putting forward a general strategy for designing ARX-based symmetric-key primitives with provable security against single-characteristic differential and

linear cryptanalysis. As a dual to the wide trail strategy [51, 52] adopted by many S-box based block ciphers, the designers proposed the long trail strategy. This strategy promotes the use of a rather weak but large S-box, i.e., an ARX-based S-box, along with a very light linear layer. Fostering the existence of long trails, that involve an uninterrupted sequence of calls to the S-box interleaved with key additions, rather than having maximum diffusion in each linear layer is at the core of this proposed strategy. The long trail strategy allowed the designers to bound the maximum differential and linear probabilities for any number of rounds of a block cipher designed following such strategy. SPARX-64/128 is a member of this family of block ciphers following the long trail strategy with 64-bit block size and 128-bit key. The only cryptanalysis of SPARX was done by its designers as they presented a 13-round bit-based division property distinguisher that they used to launch an integral attack against 15-round SPARX-64/128 [63]. No other attacks were given in the short/full versions of the design paper.

In this chapter, we present a 12-round truncated Impossible Differential on SPARX-64/128 that can be extended to a 13-round impossible differential with a specific input difference and a truncated output difference. We use the 12-round impossible differential to launch an impossible differential attack against 15-round SPARX-64/128 including the post-whitening key with data complexity of 2^{51} chosen plaintexts, time complexity of $2^{94.1}$ 15-round encryptions and memory complexity of $2^{43.5}$ 64-bit blocks. Then, we use the 13-round impossible differential to attack 16-round SPARX-64/128, including the post-whitening key, starting from round 2 with data, time and memory complexities of $2^{61.5}$ known plaintexts, 2^{94} 16-round encryptions, and $2^{61.5}$ 64-bit blocks, respectively.

The remainder of this chapter is organized as follows. In Section 6.2, the notation used throughout this chapter are given followed by the specification of SPARX-64/128. Our impossible differentials are presented in Section 6.3. Afterwards, in Section 6.4, we provide a detailed description of our impossible differential attacks on SPARX-64/128. Finally, Section 6.5 summarizes the chapter.

6.2 Description of SPARX-64/128

The following notations are used throughout this chapter:

- K: The master key.
- k_i: The i^{th} 16-bit of the key state, where $0 \leq i \leq 7$.
- k_i^j: The i^{th} 16-bit of the key state after applying the key schedule permutation j times, where $0 \leq i \leq 7$ and $0 \leq j \leq 17$ for SPARX-64/128.
- $RK_{(a,i)}$: The 32-bit round key used at branch a of round i where $0 \leq i \leq 24$ and $a = 0$ (1) denotes the left (right) branch of SPARX-64/128.
- $X_{(a,i)}$ ($Y_{(a,i)}$): The left (right) 16-bit input at branch a of round i where $0 \leq i \leq 24$, $a = 0$ (1) denotes the left (right) branch of SPARX-64/128, and the LSB of either $X_{(a,i)}$ or $Y_{(a,i)}$ is on the right.
- w: The number of 32-bit words, i.e., $w = 2$ for a 64-bit block and $w = 4$ for a 128-bit master key.
- R^3: The iteration of three rounds of SPECKEY with their corresponding key additions.
- L_w: Linear mixing layer used in SPARX with w-word block size, thus L_2 represents the linear mixing layer used in SPARX-64/128.
- $\boxplus$: Addition mod 2^{16}.
- $\oplus$: Bitwise XOR.
- $\lll q$ ($\ggg q$): Rotation of a word by q bits to the left (right).
- $\|$: Concatenation of bits.
- $0xabcd$: A 16-bit number in hexadecimal representation.

6.2.1 Specifications of SPARX-64/128

SPARX [62, 63] is a family of ARX-based SPN block ciphers. It follows the SPN design strategy while using ARX-based S-boxes instead of S-boxes based on look-up tables. ARX-based S-boxes form a specific category of S-boxes that rely solely on addition, rotation and XOR operations to provide both confusion (non-linearity) and diffusion. The SPARX family adopts the 32-bit SPECKEY ARX-based S-box, shown in Fig. 6.1, which resembles one round of SPECK-32 [15, 16] with only one difference, that is, the key is added to the whole 32-bit state instead of just half the state as in SPECK-32.

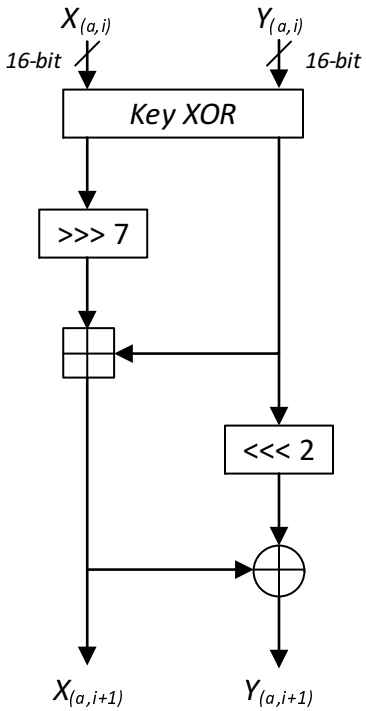

Figure 6.1: The SPECKEY ARX-based S-box used in the SPARX family.

For a given member of the SPARX family whose block size is n bits, the plaintext is divided into $w = n/32$ words of 32 bits each. Then, the SPECKEY S-box (S), being applied to w words in parallel, is iterated r times interleaved by the addition of independent subkeys. Then, a linear mixing layer (L_w) is applied to ensure diffusion between the words. The structure made of a key addition followed by S is called a round while the structure made of r rounds followed by L_w

is called a step, as depicted in Fig. 6.2. Thus, the ciphertext corresponding to a given plaintext is generated by iterating such steps. The number of steps and the number of rounds in each step depend on both the block size of the cipher and the size of the key it utilizes.

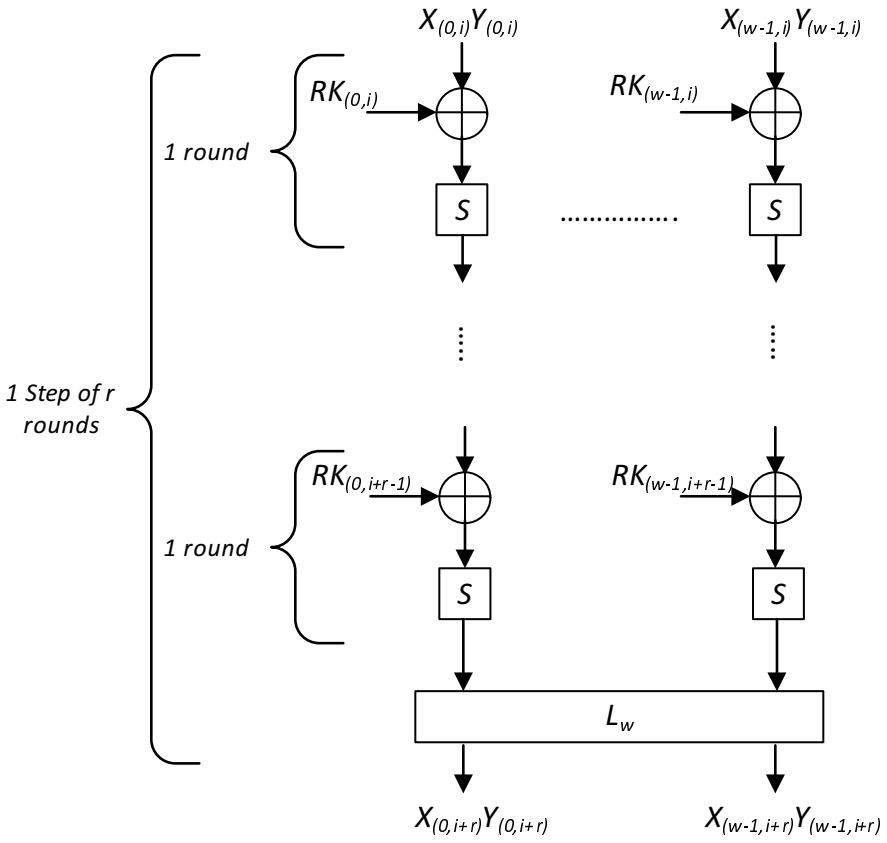

Figure 6.2: SPARX structure

SPARX-64/128 is the lightest member of this family operating on 64-bit blocks using 128-bit keys. It uses three rounds in each step and iterates over eight steps, i.e., the total number of rounds is 24. More precisely, in SPARX-64/128, 2 SPECKEY S-boxes (S) are iterated simultaneously 3 times, while being interleaved by the addition of the round keys and then a linear mixing layer (L_2) is applied, as shown in Fig. 6.3. The structure of L_2 is depicted in the dotted square in Fig. 6.4.

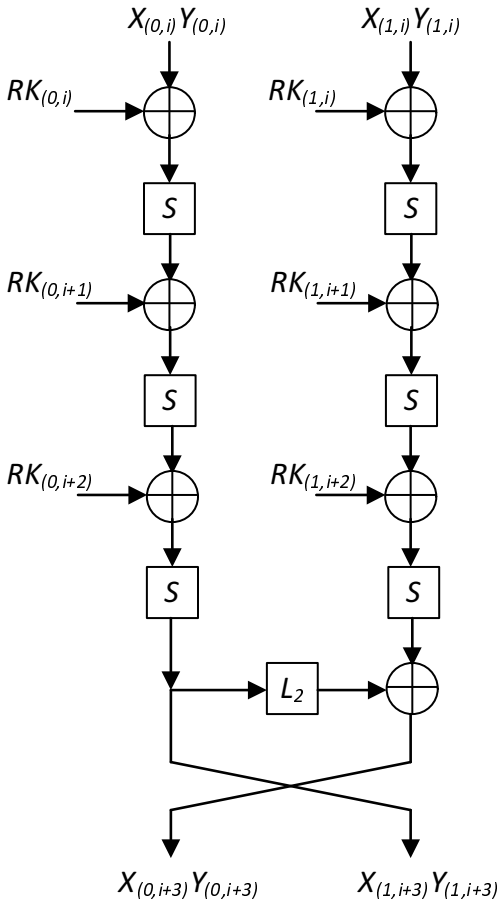

Figure 6.3: One step of SPARX-64/128

Key schedule. The 128-bit master key instantiates the key state, denoted by $k_0^0 \| k_1^0 \| k_2^0 \| k_3^0 \| k_4^0 \| k_5^0 \| k_6^0 \| k_7^0$. Then, the 3×32-bit round keys used in the left branch of the first step are extracted. Afterwards, the permutation illustrated in Fig. 6.5 is applied and then the 3×32-bit round keys used in the right branch of the first step are extracted. The application of the permutation and the extraction of the keys are interleaved until all the round keys encompassing the post-whitening ones are generated. This means that, first, the round keys of a branch of a given step j are generated and then the key state is updated. The following observation on the key schedule is exploited in our attacks.

Observation: The last round key of a given step and the first round key of the subsequent step can be deduced from one another. To clarify this point, we consider the last round key of step

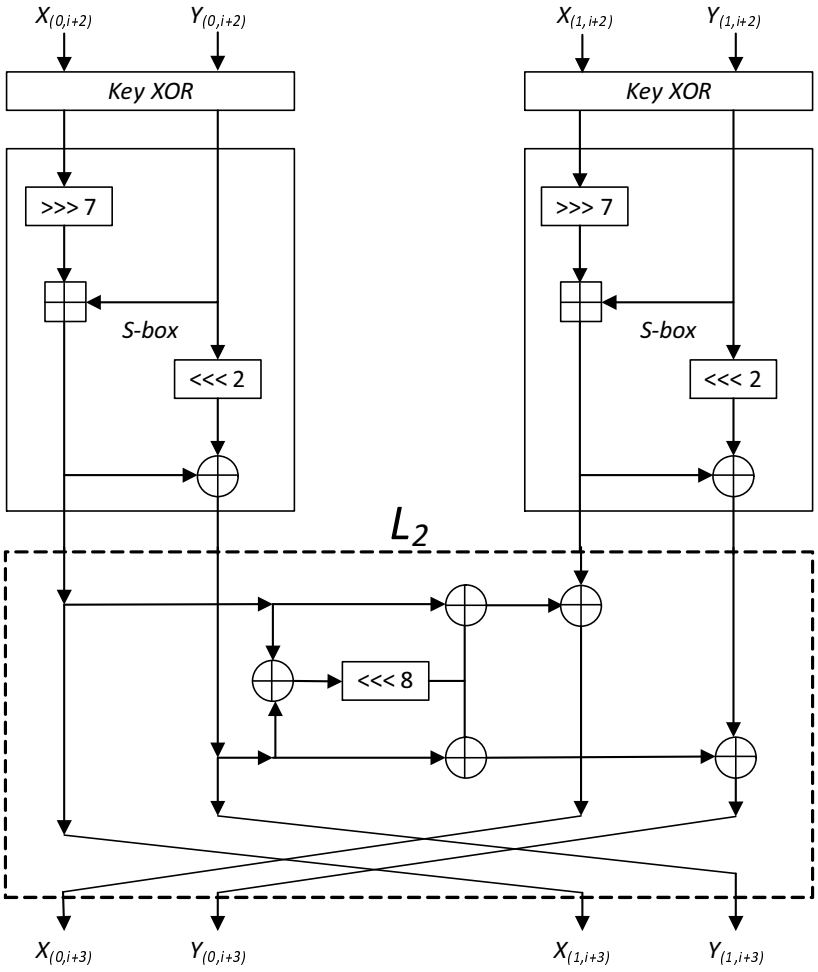

Figure 6.4: One round and linear layer of SPARX-64/128

zero and the first round key of step one. The 64-bit round key of the third round is $k_4^0 \| k_5^0, k_4^1 \| k_5^1$ and the 64-bit round key of the fourth round is $k_0^2 \| k_1^2, k_0^3 \| k_1^3$. According to the key schedule: $k_0^2 = k_6^1 = k_4^0$, $k_1^2 = k_7^1 \boxplus 2 = k_5^0 \boxplus 2$, $k_0^3 = k_6^2 = k_4^1$ and $k_0^3 = k_7^1 \boxplus 3 = k_5^1 \boxplus 3$.

Finally, it is to be noted that we measure the memory complexity of our attacks in number of 64-bit blocks and the time complexity in terms of the equivalent number of round-reduced encryptions.

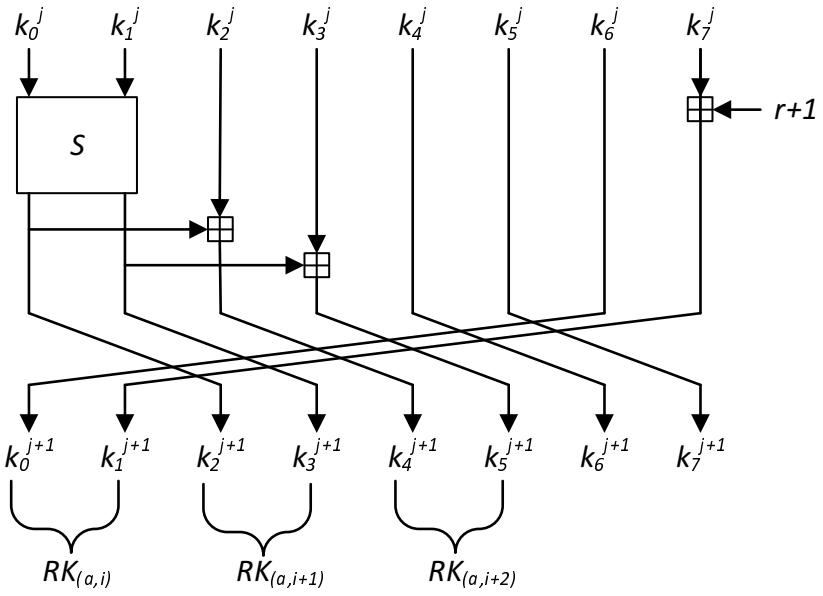

Figure 6.5: SPARX-64/128 key schedule permutation, where the counter r is initialized to zero.

6.3 Impossible Differentials of SPARX-64/128

A 12-round impossible differential is readily noticeable when considering SPARX-64/128 to be a twisted variant of a Feistel construction where the two halves undergo a keyed function before getting mixed and swapped. Indeed, as depicted in Fig. 6.6, if the left branch of SPARX-64/128 at round i has a zero difference while the right half has a nonzero difference, then after 2 steps (6 rounds), the input at the left branch must have a nonzero difference. From the other direction, if the input of the right branch of round $i + 12$ has a nonzero difference, i.e, Γ and the input of the left branch at that round has a difference $L_2(\Gamma)$, then after the linear transformation, the right branch will have a zero difference which propagates unaltered for 2 complete steps (6 rounds) and contradicts with the forward differential at the left branch.

This 12-round truncated impossible differential can be extended to a 13-round distinguisher with a specific input difference and truncated output difference. This is feasible by exploiting the fact that there exist differentials with probability 1 for one SPECKEY round and one of these differentials is a fixed point of L_2. Particularly, if the input difference of the distinguisher

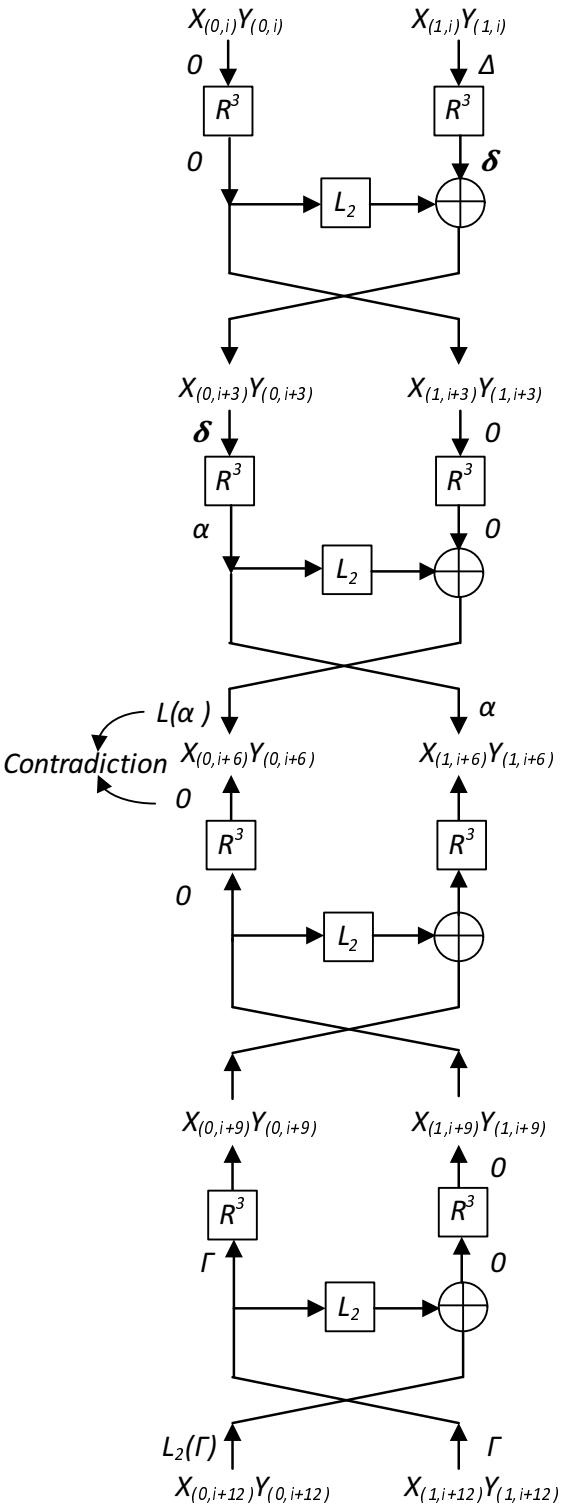

Figure 6.6: 12-round impossible differential SPARX-64/128

is chosen to be $0x8000\ 0x8000$ then by propagating it backward through L_2 we have the same difference at both the right and left branches as an output for the S-box and this output difference corresponds to the input difference $0x0040\ 0x0000$ with probability 1. Hence, the input of the 13-round distinguisher is $0x0040\ 0x0000$ and $0x0040\ 0x0000$ while the output is still truncated in the form of $L_2(\Gamma)$ and Γ.

6.4 Impossible Differential Cryptanalysis of SPARX-64/128

The 12 and 13-round impossible distinguishers described above can be used to attack 15 and 16-round SPARX-64/128, respectively. Both attacks include the post-whitening key, however, the 16-round attack starts at round two.

6.4.1 15-round Impossible Differential Attack on SPARX-64/128

In this attack, we have chosen to place the 12-round distinguisher at the top, end it with a specific difference that meets the constraint of $L_2(\Gamma)$ and Γ, and then append 3 rounds that have a high probability as shown in Fig. 6.7. That specific difference at the end of the distinguisher and the three analysis rounds were found using MILP. More specifically, we have followed the guidelines in [69] to create an MILP model that describes SPARX-64/128 and solved it using the publicly available MILP optimizer Gurobi [71]. The detailed procedure of the attack is described as follows.

Data Collection. We first choose 2^m structures of plaintexts where in each structure the left 32 bits of the plaintexts take a fixed value and the right 32 bits take all the 2^{32} possible values. Each structure includes about $\binom{2^{32}}{2} \approx 2^{63}$ pairs of plaintexts, therefore we have $2^m \times 2^{63} = 2^{m+63}$ pairs of plaintexts in total. We encrypt these pairs and keep the ones whose ciphertext difference matches the difference shown in Fig. 6.7. The probability of such ciphertext difference is about 2^{-64}, therefore the expected number of remaining pairs after this phase is about $2^{m+63-64} =$

2^{m-1}.

Key Recovery. To verify if the pairs generated during the data collection phase follow our 12-round impossible differential, we need to guess $RK_{(0,15)}, RK_{(1,15)}, RK_{(0,14)}, RK_{(1,14)}$, and $RK_{(0,13)}$. However, as pointed out above, $RK_{(0,15)}$, $RK_{(1,15)}$ are related to $RK_{(0,14)}, RK_{(1,14)}$. This means that these round keys take 2^{96} values only. The details of this phase are as follows.

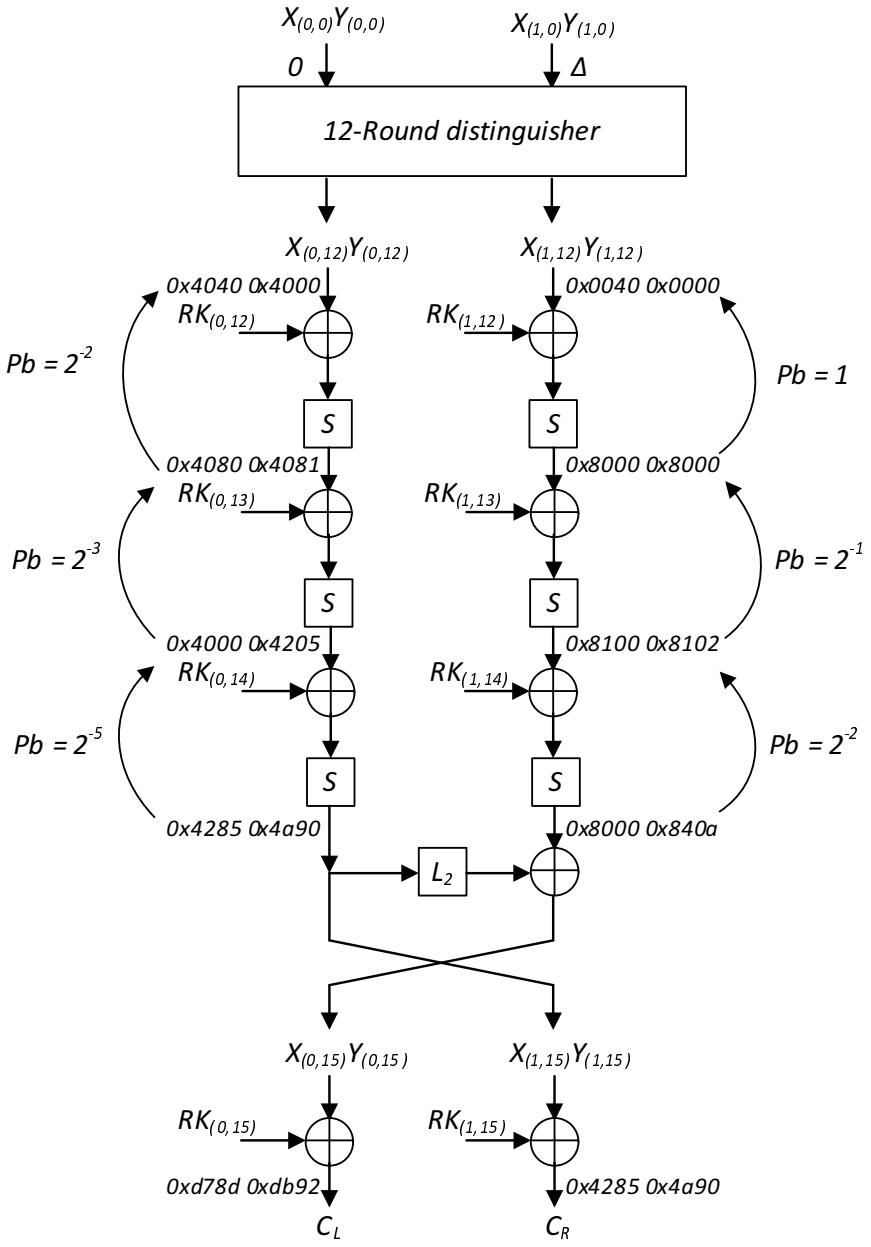

Figure 6.7: 15-round impossible differential attack on SPARX-64/128

Step 1. For all the ciphertext pairs obtained in the data collection phase, we guess the 64-bit round keys $RK_{(0,15)}$ and $RK_{(1,15)}$, decrypt round 15 and check if the difference matches the one shown in Fig. 6.7. If it is not the case, the pair is discarded. The probability of this event is 2^{-7} and thus after this step the expected number of remaining pairs is about $2^{m-1-7} = 2^{m-8}$.

Step 2. We deduce $RK_{(0,14)}$ and $RK_{(1,14)}$ from the guessed $RK_{(0,15)}$ and $RK_{(1,15)}$, decrypt round 14 and check if the difference is the expected one according to Fig. 6.7. If it is not the case, the pair is discarded. The probability of this event is 2^{-4} and therefore the expected number of pairs surviving this step is about $2^{m-8-4} = 2^{m-12}$.

Step 3. We guess the 32-bit $RK_{(0,13)}$ and partially decrypt the left branch of round 13 and check if the difference meets the impossible differential difference. Once it is correct, we delete the 32-bit round key guesses of $RK_{(0,13)}$ since such a differential is impossible; each round key guess that proposes such a difference is a wrong key. After analyzing all the 2^{m-12} remaining pairs, we output the 96-bit round keys guess of $RK_{(0,15)}$, $RK_{(1,15)}$, and $RK_{(0,13)}$ as a candidate. The probability that the pairs pass this step is about 2^{-2}, therefore the time complexity of this step is the number of key guesses $\times$ 2 messages in each pair $\times$ the probability that the key guess is excluded after sequentially testing it against all the surviving pairs.

The steps of the key recovery phase are described in Table 6.1, whereas the second column gives the round keys to be guessed in the corresponding round for each attack step. The third column presents the number of surviving pairs after each step, and the fourth column is the time complexity of each step measured in 15-round encryption.

Attack complexity. To balance the attack complexity between the different phases, we set $m = 19$. This means that after analyzing all the remaining pairs, there will be about $2^{96} \times (1 - 2^{-2})^{2^{m-12}} = 2^{96} \times (0.75)^{128} \approx 2^{42.9}$ remaining candidates for the 96-bit round keys. Then, we

Attack step	Guessed keys	# Surviving pairs	Time complexity
1	$RK_{(0,15)}$ $RK_{(1,15)}$	$2^{m-1-7} = 2^{m-8}$	$2^{64} \times 2 \times 2^{m-1} \times 1/15 \approx 2^{m+60.1}$
2	†	$2^{m-8-4} = 2^{m-12}$	$2^{64} \times 2 \times 2^{m-8} \times 1/15 \approx 2^{m+53.1}$
3	$RK_{(0,13)}$	–	$2^{96} \times 2 \times [1 + (1 - 2^{-2}) + (1 - 2^{-2})^2$ $+ \cdots + (1 - 2^{-2})^{2^{m-12}}] \times 1/(2 \times 15)$

Table 6.1: Key recovery process of the attack on 15-round SPARX-64/128. †: No additional key guesses needed, i.e., the round keys are deduced from the previously guessed ones.

guess the 32-bit $RK_{(1,12)}$ which along with the surviving candidates allows us to recover the master key K via the key schedule. Afterwards, we test each one of these master key candidates using two plaintext/ciphertext pairs to find the correct master key. The time complexity of this exhaustive search step is $2 \times 2^{32} \times 2^{42.9} = 2^{75.9}$. Therefore the time complexity is dominated by step 3 of the attack and estimated to be $2^{96} \times 2 \times (1/2^{-2}) \times (1/30) \approx 2^{94.1}$. The data complexity of the attack is $2^{19+32} = 2^{51}$ chosen plaintexts. The memory complexity of the attack is dominated by the memory that is required to store the keys to be excluded, i.e., $2^{42.9} \times 96/64 \approx 2^{43.5}$ 64-bit blocks.

6.4.2 16-round Impossible Differential Attack on SPARX-64/128

Although each round of SPARX-64/128 uses a 64-bit round key, there exists three specific rounds that contain only 2^{96} bits of key information as exemplified by the ones exploited in the previous attack. Nonetheless, any four rounds contain at least 128 bits of key information. Therefore, our 16-round attack on SPARX-64/128 has to start from round two and in this case, we use the 13-round impossible differential and prepend 3 rounds on its top as shown in Fig. 6.8. Again, we have used the Gurobi optimizer to find these three rounds after creating the MILP model that describes them.

In this attack, we do not use data structures as they do not generate enough pairs to launch the attack. Instead, we use known plaintexts and generate the pairs we need probabilistically.

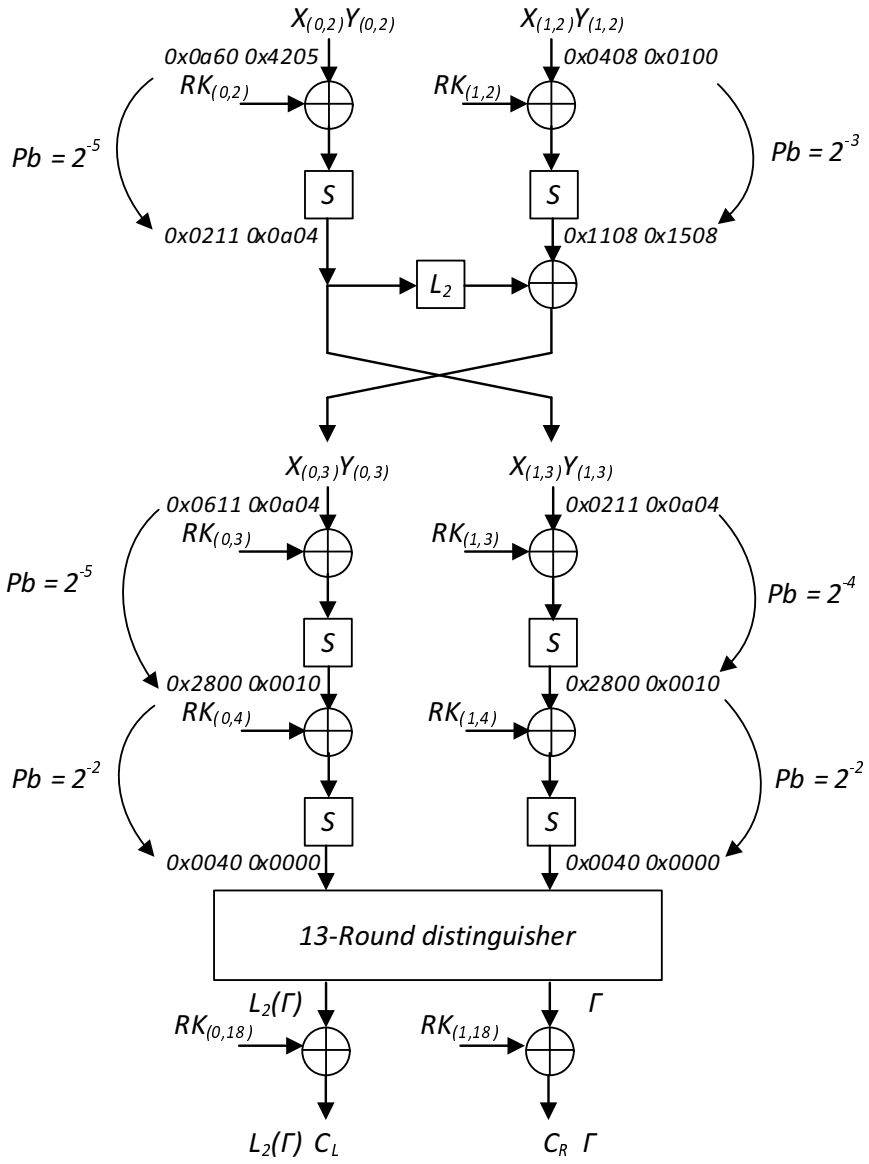

Figure 6.8: 16-round impossible differential attack on SPARX-64/128

Hence, if we have $2^{61.5}$ known plaintexts, these can generate $\binom{2^{61.5}}{2} \approx 2^{122}$ pairs. Out of these pairs, we would have $2^{122-64} = 2^{58}$ pairs that satisfy the plaintext difference shown in Fig. 6.8. Then, as the difference at the end of the distinguisher is the difference in the ciphertext, we have to filter the ciphertexts such as the right branch is a nonzero difference Γ and the left branch difference is $L_2(\Gamma)$ which means that we have $2^{58-32} = 2^{26}$ proper pairs.

In the key recovery phase, which we perform on these 2^{26} pairs, the 3 round keys take 2^{96} values only and they are guessed on steps to reduce the time complexity of the attack as listed in Table 6.2. It is to be noted that, according to the key schedule, $RK_{(0,3)}$, $RK_{(1,3)}$ are deduced from the guessed $RK_{(0,2)}$, $RK_{(1,2)}$ and that $RK_{(1,4)}$ is deduced from $RK_{(0,3)}$.

Attack step	Guessed keys	# Surviving pairs	Time complexity
1	$RK_{(0,2)}$ $RK_{(1,2)}$	$2^{26-8} = 2^{18}$	$2^{64} \times 2 \times 2^{26} \times 1/16 = 2^{87}$
2	†	$2^{18-9} = 2^{9}$	$2^{64} \times 2 \times 2^{18} \times 1/16 = 2^{79}$
3	†	$2^{9-2} = 2^{7}$	$2^{64} \times 2 \times 2^{9} \times 1/(2 \times 16) = 2^{69}$
4	$RK_{(0,4)}$	–	$2^{96} \times 2 \times [1 + (1 - 2^{-2}) + (1 - 2^{-2})^2$ $+ \cdots + (1 - 2^{-2})^{2^7}] \times 1/(2 \times 16)$

Table 6.2: Key recovery process of the attack on 16-round SPARX-64/128. †: No additional key guesses needed, i.e., the round keys are deduced from the previously guessed ones.

After analyzing all the remaining pairs, there will be about $2^{96} \times (1 - 2^{-2})^{2^7} = 2^{96} \times (0.75)^{128} \approx 2^{42.9}$ remaining candidates for the 96-bit round keys. Then, we guess the remaining 32 bits of the master key and test each one of these master key candidates using 2 plaintext/ciphertext pairs to find the correct one. The time complexity of this exhaustive search step is $2 \times 2^{32} \times 2^{42.9} = 2^{75.9}$. Therefore the time complexity is dominated by step 4 of the attack (see Table 6.2) and estimated to be $2^{96} \times 2 \times (1/2^{-2}) \times (1/32) = 2^{94}$. The data complexity of the attack is $2^{61.5}$ known plaintexts. In this case, the memory complexity of the attack is dominated by the hash table [107] that is used to store the plaintexts while generating the required pairs, i.e., $2^{61.5}$ 64-bit blocks.

6.5 Summary

In this chapter, we have analyzed SPARX-64/128 against the impossible differential attack. We have presented 12 and 13-round impossible differential distinguishers that are used to attack

15 and 16-round SPARX-64/128 with the post-whitening key, respectively. The (data complexity in chosen/known plaintexts, time complexity in 15/16-round encryptions, memory complexity in 64-bit blocks) of these attacks are $(2^{51}, 2^{94.1}, 2^{43.5})$ and $(2^{61}, 2^{94}, 2^{61.5})$, respectively. This is the first third-party cryptanalysis of SPARX-64/128.

Chapter 7

Impossible Differential Cryptanalysis of 8-round Kiasu-BC

In ASIACRYPT 2014, TWEAKEY, a framework for unifying the design of tweakable block ciphers was proposed. Kiasu-BC is one of the instantiations of this framework, which can be considered as tweakable AES-128. It reuses the same round function and key schedule of AES-128, the only difference is the addition of a 64-bit tweak to the first two rows of every round key. In their security analysis, the designers concluded that the security of Kiasu-BC is identical to AES-128 when it comes to impossible differential attacks and truncated differential attacks. In this chapter, we show that an 8-round impossible differential attack on Kiasu-BC is feasible. We adopt one of the existing impossible differential distinguishers on AES-128 and exploit the tweak to gain this extra round in comparison to the impossible differential attacks on AES-128.

7.1 Introduction

The notion of tweakable block ciphers was first proposed by Liskov, Rivest and Wagner [101]. In addition to the usual plaintext and secret key inputs of conventional block ciphers, a tweakable block cipher accepts a third input called the "tweak". The tweak is public and provides variability similar to the role of an initialization vector in the cipher block chaining

(CBC) mode or a nonce in the offset codebook (OCB) mode. The design of tweakable block ciphers is normally done such that tweaks can be changed efficiently in comparison to the key setup, which makes tweakable block ciphers a good base to build secure modes of operation. The advantage of building tweakable block ciphers from conventional block ciphers is two-fold. First, the extra cost of turning a block cipher implementation into a tweakable one is small. Second, a tweakable block cipher would presumably inherent the trust and the thorough security analysis of its underlying block cipher.

The TWEAKEY framework for unifying the design of tweakable block ciphers was proposed by Jean et al. [78, 79]. They have put forward three specific instances of their framework, namely, Deoxys-BC, Joltik-BC, and Kiasu-BC. The latter is the standardized AES-128 block cipher turned into a tweakable block cipher. This is achieved by accepting, in addition to the usual inputs of AES-128, a 64-bit tweak T that is XORed to the first two rows of every round key. In other words, if T is set to zero, then Kiasu-BC is AES-128.

The impact of introducing the tweak on the security of Kiasu-BC was first studied by its designers in their authenticated encryption scheme proposal [80] submitted to the ongoing CAESAR competition [39]. The focus of their security analysis was to evaluate the security of Kiasu-BC against the related-key related-tweak differential attacks and the met-in-the-middle attacks. For the first, they showed that no 7-round differential characteristic would have a probability higher than 2^{-128}. For the latter, they concluded that the same attacks existing for AES-128 applies to Kiasu-BC. For the remaining types of attacks, they have argued that the security level of Kiasu-BC stays the same. In particular, for the impossible differential attacks, the designers stated: *"The impossible differential attacks on AES do not exploit the key schedule, neither do the truncated differential attacks, and therefore the number of rounds that they can penetrate would remain the same"* [80]. The first third-party analysis of Kiasu-BC was done by Dobraunig, Eichlseder and Mendel [66], where they studied the security of Kiasu-BC against Square attacks [50] and exploited the tweak to build a 4-round Square-based distinguisher which they

used to launch a 7-round attack on Kiasu-BC with significantly better complexities than the best published attacks on 7-round of AES-128.

The impossible differential attack was mounted on AES for the first time by Biham and Keller [24], where they presented a 4-round impossible differential distinguisher that is composed of two deterministic differentials that contradict in the middle. A notable improvement of their attack was achieved by Bahrak and Aref [13] and, independently, Wentao, Wenling, and Dengguo who extended the attack to 7-round AES-128 by using a different 4-round impossible differential distinguisher and by adding 3 analysis rounds. These attacks were improved by Lu et al. [103] by utilizing the early abort technique and exploiting the key schedule relations. They were further improved by Mala et al. [107] by presenting a different 4-round impossible differential distinguisher and making use of the redundancy in the key schedule in a more lucrative way.

In this chapter, we present an Impossible Differential attack on 8-round Kiasu-BC by carrying over the previous techniques employed in attacking AES-128 and leveraging the freedom in the choice of the tweak. Specifically, in addition to using the early abort technique, the precomputation tables and structures of plaintexts, we exploit the tweak to add one more analysis round. Our attack requires $2^{116.64}$ chosen plaintexts encrypted using two tweaks, i.e., $2^{117.64}$ chosen plaintexts-tweaks combinations and has a time complexity equivalent to $2^{118.97}$ 8-round encryptions and uses $2^{101.64}$ 128-bit memory blocks.

The rest of this chapter is organized as follows. First, we briefly describe the specifications of Kiasu-BC in Section 7.2. Then, we present our impossible differential distinguisher in Section 7.3, followed by the key-recovery attack on 8-round Kiasu-BC in Section 7.4. Finally, we summarize this chapter in Section 7.5.

7.2 A Brief Description of Kiasu-BC

Kiasu-BC is a tweakable block cipher, which uses a 64-bit tweak to define a different specific keyed instance of AES-128, i.e., if the tweak is set to 0, then Kiasu-BC is identical to AES-128. Thus, similar to AES-128, the internal state of Kiasu-BC is represented as a 4×4 matrix that undergoes 10 rounds of the following five transformations:

- SubBytes (SB) where the 8-bit AES S-box is applied simultaneously to the 16-byte internal state.
- ShiftRows (SR) where row i of the internal state is rotated left by i bytes, $0 \leq i \leq 3$.
- MixColumns (MC) where each column of the internal state is multiplied by an MDS matrix defined over the finite field $GF(2^8)$.
- AddroundKey (AK) where the internal state is XORed to the 128-bit round key RK_i generated by applying the AES-128 key schedule on the 128-bit master key.
- AddTweak (AT) where the first two rows of the internal state is XORed to the same 64-bit tweak T.

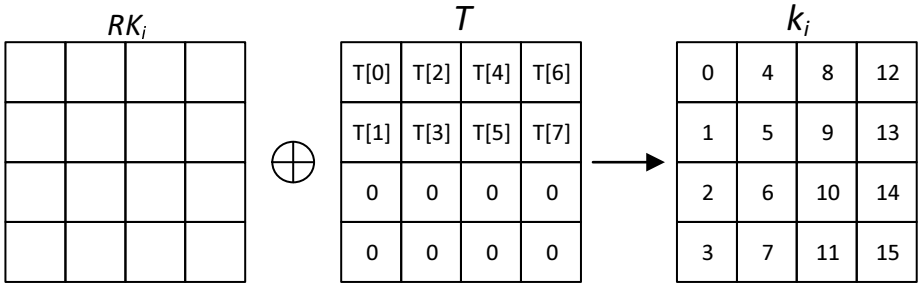

Figure 7.1: Tweak addition to the round key in Kiasu-BC

Additional AK and AT transformations are applied in the first round and the MC operation is omitted from the last round. As both the AK and AT transformations are linear, we merge them into one transformation that we refer to as AKT, where k_i is the XOR of the round key RK_i and the tweak T in round i, as illustrated in Figure 7.1.

The input of the AES-128 key schedule is the 128-bit master key and it outputs 128-bit 11 round keys. The keys are arranged in an array denoted by $W[0, \ldots, 43]$, where each word of $W[.]$ is 32 bits and corresponds to a column in the round key. The master key is placed as the first 4 words of $W[.]$. The other words of W are generated according to the following algorithm:

- For $i = 4, \ldots, 43$, do:
  - If $i \equiv 0 \bmod 4$ then $W[i] = W[i-4] \oplus SB(W[i-1] \lll 8) \oplus RCON[i/4]$, where $RCON[.]$ is an array of predetermined constants and $\lll n$ denotes rotation of the word by n bits to the left,
  - otherwise, $W[i] = W[i-1] \oplus W[i-4]$.

In this chapter, we use x_i, y_i, z_i, and w_i to denote the 16-byte internal state in round i after the AKT, SB, SR, and MC operations, respectively, where $1 \leq i \leq 10$. $x_i[j]$ denotes the j^{th} byte of x_i, where $0 \leq j \leq 15$, as shown in k_i in Figure 7.1. The bytes located in positions $j, k, l, \ldots$ of x_i are denoted by $x_i[j, k, l, ...]$. The differences in x_i and $x_i[j]$ are denoted by Δx_i and $\Delta x_i[j]$, respectively. Finally, in some cases, we are interested in interchanging the order of the MC and the AK transformations. The involved operations are linear and hence they can be interchanged by first XORing the data with an equivalent round key, applying the MC transformation and then XORing with the tweak. In these cases, $u_i = MC^{-1}(RK_i)$ is used to denote the equivalent round key.

7.3 An Impossible Differential Distinguisher of Kiasu-BC

In our attacks, we utilize messages that are encrypted using the same secret key but two different tweaks that have any arbitrary nonzero difference in byte 0 and zero difference in the other bytes. We adopt the impossible differential distinguisher that was used in [107] and make the necessary adjustments due to the presence of the tweak difference. Hence, our impossible differential distinguisher states that given a pair of w_2 with six active bytes distributed on 2

columns as shown in Figure 7.2, w_5 cannot have nonzero difference in byte 0 and zero difference in the other 15 bytes. The reason is that, even with the presence of the tweak difference, the six active bytes in w_2 will result in an entirely inactive column of w_3. From the other side, the active byte $w_5[0]$ will result in, at least, 12 active bytes in x_4 and will contradict with certainty with w_3 in bytes 4, 6 and 7 (The bytes marked with X in both w_3 and x_4 in Figure 7.2). It is to be noted that there are another two patterns for the six active bytes in w_2 that will be exploited in our attack. These two patterns will be elaborated when discussing the attack procedure and in particular the precomputation tables.

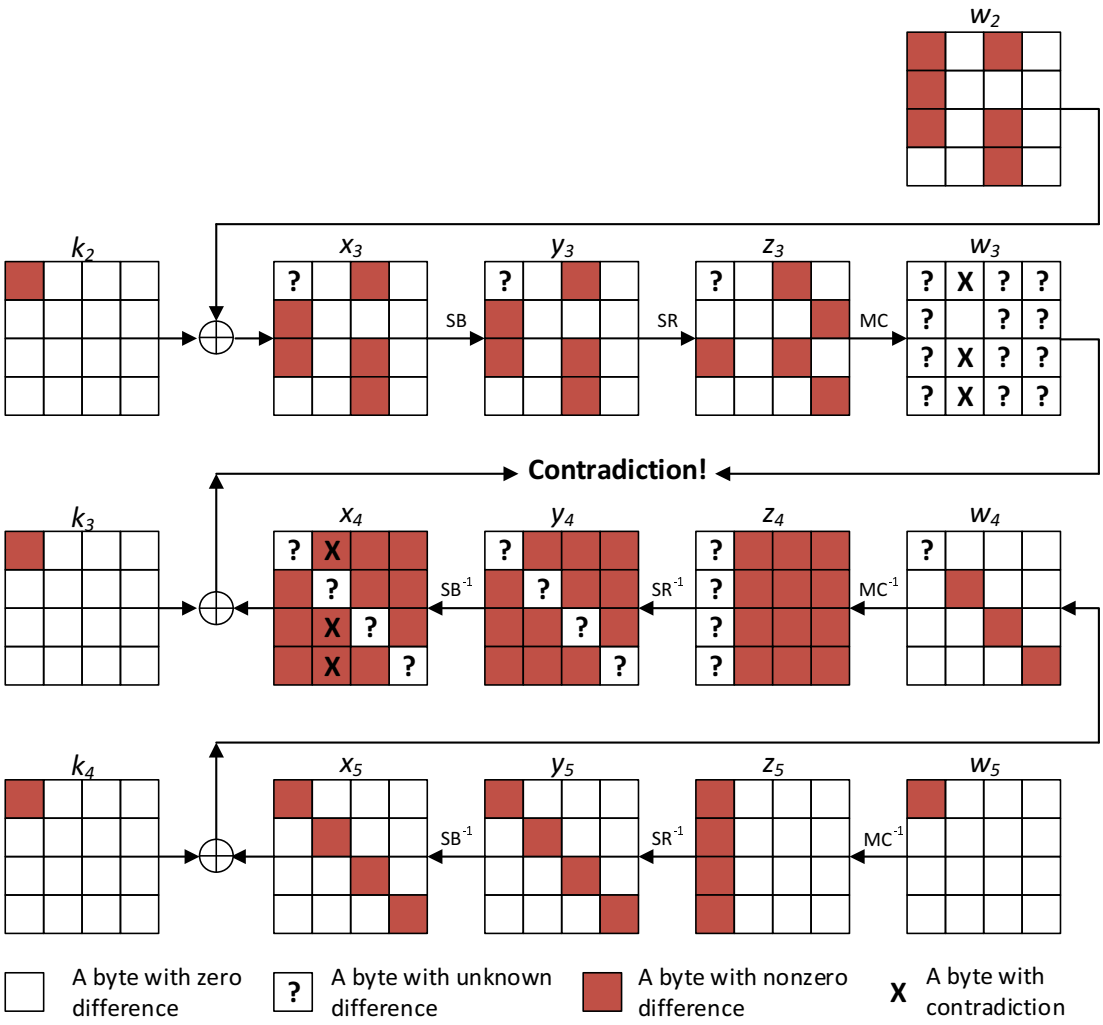

Figure 7.2: Impossible differential distinguisher of Kiasu-BC

7.4 Impossible Differential Key-recovery Attack on 8-round Kiasu-BC

We mount our attack against 8-round Kiasu-BC using the previous impossible differential distinguisher, prepending two rounds and appending three rounds, as depicted in Figure 7.3. First, we build five types of precomputation tables $H_1, H_{2,i}, H_{3,j}, H_4, H_5$ that would help extract the key bytes $k_0[2,7,8,13]$, $k_1[10]$, $k_1[8]$, $k_8[0,7,10,13]$ $u_7[0]$, respectively. These tables are built as follows:

H_1: For all the about 2^{48} possible pairs of $(w_1[8,9,10,11], w_1'[8,9,10,11])$ which have zero difference in bytes $[9,11]$ and nonzero difference in bytes $[8,10]$, compute the values of $(x_1[2,7,8,13], x_1'[2,7,8,13])$. The computed pairs are stored in H_1 indexed by $\Delta x_1[2,7,8,13]$. H_1 has 2^{32} rows and on average about $\frac{2^{48}}{2^{32}} = 2^{16}$ pairs lie in each row.

$H_{2,i}$, $i = 1,2,3$: For all the about 2^{32} possible pairs of $(x_2[0,10], x_2'[0,10])$ which have nonzero difference in these two bytes, compute $\Delta w_2[0,1,2,3]$. Then, for $i = 1,2,3$ choose the pairs $(x_2[0,10], x_2'[0,10])$ who lead to a zero difference in byte i of $\Delta w_2[0,1,2,3]$. This is an 8-bit filtration so the expected number of such pairs is $2^{32} \times 2^{-8} = 2^{24}$. These pairs are stored in $H_{2,i}$ indexed by $x_2[0] \| x_2'[0] \| \Delta x_2[10]$. Each of the three $H_{2,i}$ tables has 2^{24} rows and on average about $\frac{2^{24}}{2^{24}} = 1$ pair lie in each row.

$H_{3,j}, j = 8,9,11$: For all the about 2^{32} possible pairs of $(x_2[2,8], x_2'[2,8])$ which have nonzero difference in these two bytes, compute $\Delta w_2[8,9,10,11]$. Then, for $j = 8,9,11$ choose the pairs $(x_2[2,8], x_2'[2,8])$ who lead to a zero difference in byte j of $\Delta w_2[8,9,10,11]$. This is an 8-bit filtration so the expected number of such pairs is $2^{32} \times 2^{-8} = 2^{24}$. These pairs are stored in $H_{3,j}$ indexed by $x_2[2] \| x_2'[2] \| \Delta x_2[8]$. Each of the three $H_{3,j}$ tables has 2^{24} rows and on average about $\frac{2^{24}}{2^{24}} = 1$ pair lie in each row.

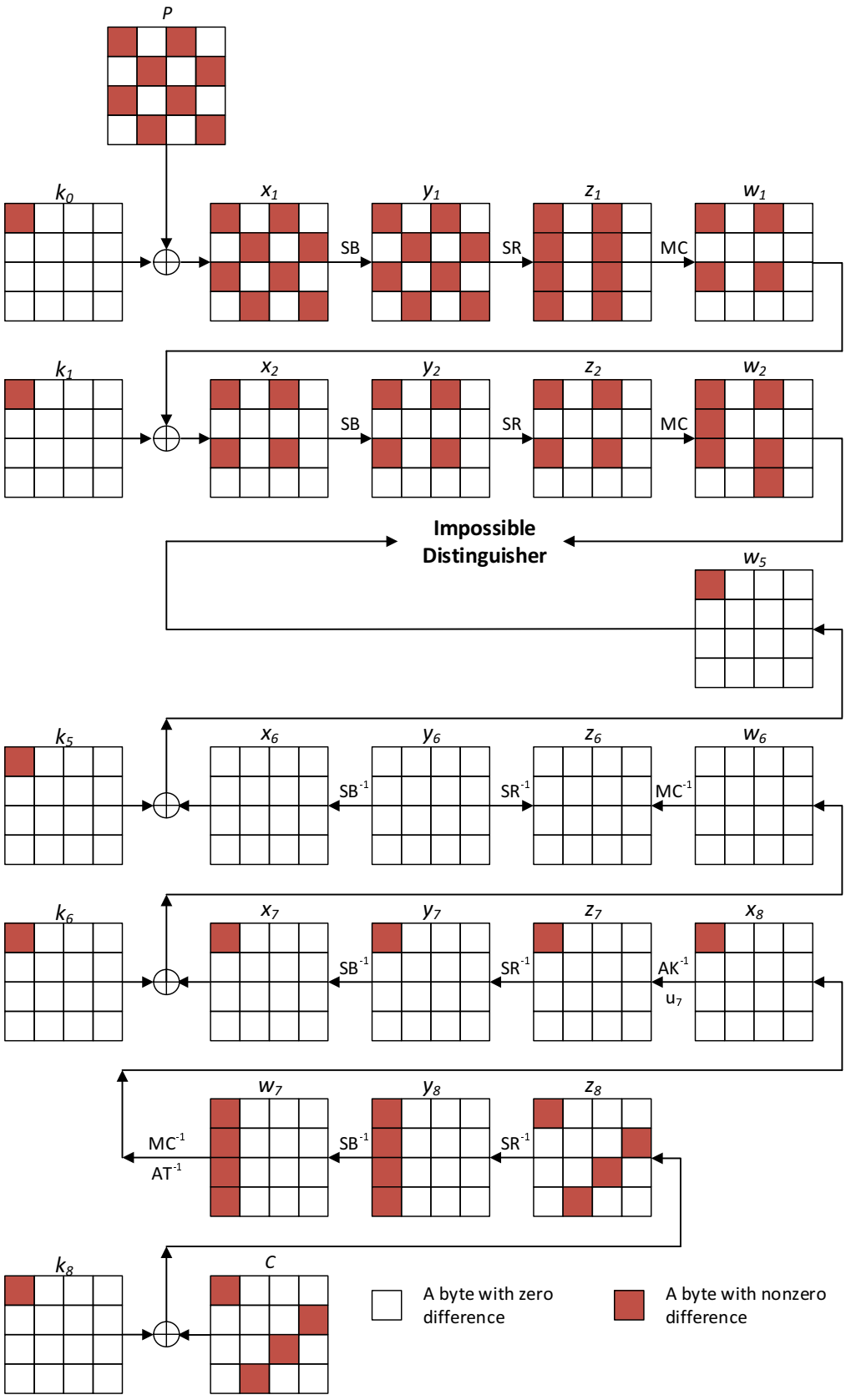

Figure 7.3: Impossible differential attack on 8-round Kiasu-BC

In the attack, each of the three tables $H_{3,j}$ can be used together with only one of the three tables $H_{2,i}$, hence we denote $H_{3,conj(i)}$ as the conjugate of $H_{2,i}$, where for $i = 1, 2, 3$, the $conj(i)$ are 11, 8, 9, respectively. In comparison to [107], we have excluded a fourth table because that table, with the presence of the tweak difference, would not lead to a deterministic impossible differential. The chosen position of the tweak difference determines which table to exclude. In our attack, the tweak difference is chosen at position 0 and therefore we do not include the table where $i = 0$, and $j = 10$.

H_4: There are $(2^8 - 1)$ possible values of $\Delta z_7[0]$, consequently, there are the same number of $\Delta x_8[0]$ values (after changing the order of the MC and the AK operations). Then, for all the $2^{32} \times (2^8 - 1) \approx 2^{40}$ possible pairs of $(x_8[0, 1, 2, 3], x'_8[0, 1, 2, 3])$, compute the values of $(z_8[0, 7, 10, 13], z'_8[0, 7, 10, 13])$. These pairs are stored in H_4 indexed by $\Delta z_8[0, 7, 10, 13]$. H_4 has 2^{32} rows and on average about $\frac{2^{40}}{2^{32}} = 2^8$ pairs lie in each row. Here, we restrict the position of Δz_7 to match the position of the tweak difference.

H_5: To follow our chosen differential path, Δx_7 has one possible value only, i.e., the difference in the tweak. So, for all the 2^8 possible pairs of $(x_7[0], x'_7[0])$, compute the values of $(z_7[0], z'_7[0])$. These pairs are stored in H_5 indexed by $\Delta z_7[0]$. H_5 has 2^8 rows and on average about $\frac{2^8}{2^8} = 1$ pair lie in each row.

To generate the data pairs needed to launch the attack, we use structures of data where in each structure the bytes $(0, 2, 5, 7, 8, 10, 13, 15)$ take all the 2^{64} possible values and the other 8 bytes take arbitrary fixed value. We encrypt each structure using two tweaks T_1 and T_2 where ΔT is nonzero at byte 0 and zero elsewhere. The pairs used in the attack are generated such that one message is encrypted using T_1 and the other is encrypted using T_2. This means that, for each two structures, we can generate $2^{64} \times (2^{64} - 1) \times 1/2 \approx 2^{127}$ plaintexts-tweaks pairs [1], satisfying the

[1]The probability that $\Delta T[0]$ equals $\Delta P[0]$ is 2^{-8}, and thus the possibility they do not cancel each other is $1 - 2^{-8} \approx 1$.

conditions that bytes $(0, 2, 5, 7, 8, 10, 13, 15)$ have a nonzero difference, the remaining 8 bytes have a zero difference, and the tweak have a nonzero difference in byte 0, as shown in Figure 7.3.

The attack proceeds as follows:

1. First, we take 2^n structures encrypted using T_1 and T_2 and filter their corresponding ciphertexts such that they have a nonzero difference in bytes 0, 7, 10, 13, and a zero difference in the other 12 bytes. This is a 96-bit filtration on all the 2^{n+127} plaintexts-tweaks combinations pairs, so at the end of this step the expected number of remaining pairs from all structures is $2^{n+127} \times 2^{-96} = 2^{n+31}$ pairs.

2. Next, we use proposition 4.1 of bijective S-boxes: for specific input and output differences of the S-box, there is, on average, one pair of actual values that satisfies these differences. As $\Delta w_1[0, 1, 2, 3]$ takes $(2^8 - 1)^2 \approx 2^{16}$ possible values then so is $\Delta y_1[0, 5, 10, 15]$. Since for any of the remaining plaintext pairs, ΔP is known, Δk_0 is the known difference in the tweak, the knowledge of $\Delta y_1[0, 5, 10, 15]$ allows us to deduce the value of $k_0[0, 5, 10, 15]$ using the aforementioned S-box property. Therefore, we execute the following steps:

   - We initialize 2^{32} empty lists, each corresponds to a different guess of $k_0[0, 5, 10, 15]$.
   - For each of the 2^{n+31} remaining plaintext pairs, and for each of the 2^{16} possible values of $\Delta y_1[0, 5, 10, 15]$, we deduce the keys that lead this specific pair to that specific difference and add this pair to the list that corresponds to the deduced key guess.

     There are $2^{n+31} \times 2^{16} = 2^{n+47}$ suggestions distributed over 2^{32} possible subkeys, and hence, for each key guess, we expect 2^{n+15} pairs to be stored in the corresponding list.

   For each of the possible values of $k_0[0, 5, 10, 15]$, we access the corresponding list and for each of the 2^{n+15} pairs in that list, we perform the following:

3. We access the row with index $\Delta P[2, 7, 8, 13]$ in H_1. For each pair in that row, we deduce a candidate for $k_0[2, 7, 8, 13]$. Based on the structure of H_1, we expect 2^{16} such candidates.

4. For each of these 2^{16} candidates for $k_0[2, 7, 8, 13]$, we perform the following steps:

   (a) According to the key schedule, deduce the two bytes $k_1[0], k_1[2]$ from the knowledge of $(k_0[0], k_0[13])$, $(k_0[2], k_0[15])$, respectively.

   (b) We partially encrypt the plaintext pair to get $(x_2[0, 2], x'_2[0, 2])$ and $\Delta x_2[8, 10]$.

   (c) For $i = 1, 2, 3$:

      i. We access the row with index $x_2[0]\|x'_2[0]\|\Delta x_2[10]$ in $H_{2,i}$. For each pair in that row, we deduce a candidate for $k_1[10]$. Based on the structure of $H_{2,i}$, we expect just one such candidate.

      ii. We access the row with index $x_2[2]\|x'_2[2]\|\Delta x_2[8]$ in $H_{3,conj(i)}$. For each pair in that row, we deduce a candidate for $k_1[8]$. Based on the structure of $H_{3,conj(i)}$, we expect one such candidate.

5. We access the row with index $\Delta C[0, 7, 10, 13] \oplus \Delta T$ in H_4. For each pair in that row, we deduce a candidate for $k_8[0, 7, 10, 13]$. Based on the structure of H_4, we expect 2^8 such candidates.

6. For each of the 2^8 candidates for $k_8[0, 7, 10, 13]$, we perform the following:

   (a) After changing the order of the MC and AK operations in round 7, we partially decrypt the cipher text pair to get $(x_8[0], x'_8[0])$.

   (b) We access the row with index $\Delta x_8[0]$ in H_5. For each pair in that row, we deduce a candidate for $u_7[0]$. Based on the structure of H_5, we expect one such candidate.

7. In this step, for each of the 2^{n+15} corresponding pairs, we know $2^{16} \times 3 \times 2^8 \times 1 \approx 2^{25.58}$ joint values of $k_0[2, 7, 8, 13]\|k_1[0, 2, 8, 10]\|k_8[0, 7, 10, 13]\|u_7[0]$ that result in the impossible differential. We remove these values from the list of all the 2^{88} possible values for

these joint subkeys (recall that $k_1[0], k_1[2]$ are computed via the key schedule as explained in step 4 (a) and thus these 13 key bytes take 2^{88} possible values only). After trying all the pairs, if the list is not empty, we output the values in the list along with the guess of $k_0[0, 5, 10, 15]$ as the candidates for the 120-bit target subkey.

Attack Complexity For each of the 2^{32} values of $k_0[0, 5, 10, 15]$, and for each of the 2^{n+15} pairs, we remove, on average, $2^{25.58}$ values out of the 2^{88} possible values of the target subkeys. Thus the probability that a wrong subkey is not discarded with one pair is $1-\frac{2^{25.58}}{2^{88}} = 1-2^{-62.42}$. Therefore, after examining all the 2^{n+15} pairs, we would have $2^{120}(1 - 2^{-62.42})^{2^{n+15}} \approx 2^{120} \times 2^{-1.4\times 2^{n-47.42}}$ remaining candidates for the 120-bit target subkey. To have one candidate, n will be 53.84.

In Table 7.1, we calculate the time complexities of the different steps of the attack as a function of n. As can be seen from the table, the attack time complexity is dominated by steps 1 and 7. In order to optimize the overall time complexity of the attack we set $n = 52.64$, in this case, we recover 2^{68} candidates for the 120-bit target subkeys. For each of these candidates and considering the key schedule relations, two additional key bytes $k_0[4]$ and $k_0[6]$ can be determined by the knowledge of $(k_0[8], k_1[8], k_1[0])$ and $(k_0[10], k_1[10], k_1[2])$, respectively. This means that we have 2^{68} candidates for 10 bytes of k_0 and to recover the master key, we exhaustively search through the remaining 6 bytes. Hence, the overall time complexity of the attack to recover the master key is $2^{52.64+64+1} + (2^{68} \times 2^{48}) \approx 2^{118.04}$ encryptions in addition to $2^{52.64+72.58} = 2^{125.22}$ memory access. According to [53], an AES round may be implemented in a 32-bit architectures using 20 memory accesses; 16 table lookups for SB, SR, MC, and 4 table lookups for AK, therefore the overall time complexity of the attack is about $2^{118.04} + 2^{125.22} \times \frac{1}{8} \times \frac{1}{20} \approx 2^{118.97}$ encryptions. In this case, the data complexity of the attack is about $2^{116.64}$ chosen plaintexts encrypted using two tweaks, i.e., $2^{117.64}$ plaintexts-tweaks combinations.

Step	Time Complexity	n = 52.64
1	2^{n+64+1} E	$2^{117.64}$
2	$2^{n+31} \times 2^{16} = 2^{n+47}$ MA	$2^{99.64}$
3	$2^{32} \times 2^{n+15} \times 2^{16} = 2^{n+63}$ MA	$2^{115.64}$
4(b)	$2^{32} \times 2^{n+15} \times 2^{16} \times \frac{8+2}{16} \times \frac{1}{8} = 2^{n+59.32}$ E	$2^{111.96}$
4(c)i	$2^{32} \times 2^{n+15} \times 2^{16} \times 3 = 2^{n+64.58}$ MA	$2^{117.22}$
4(c)ii	$2^{32} \times 2^{n+15} \times 2^{16} \times 3 = 2^{n+64.58}$ MA	$2^{117.22}$
5	$2^{32} \times 2^{n+15} \times 2^{8} = 2^{n+55}$ MA	$2^{107.64}$
6(a)	$2^{32} \times 2^{n+15} \times 2^{8} \times \frac{4}{16} \times \frac{1}{8} = 2^{n+50}$ E	$2^{102.64}$
6(b)	$2^{32} \times 2^{n+15} \times 2^{8} \times 1 = 2^{n+55}$ MA	$2^{107.64}$
7	$2^{32} \times 2^{n+15} \times 2^{16} \times 3 \times 2^{8} \times 1 = 2^{n+72.58}$ MA	$2^{125.22}$

Table 7.1: Time complexity of the different steps of the attack on 8-round Kiasu-BC, where E: Encryption and MA: Memory Access

The memory requirement of the attack is dominated by the memory needed to produce 2^{32} lists of step 2, which is equal to $2^{n+31} \times 2^{16} \times 4 = 2^{101.64}$ 128-bit blocks of memory, and the memory needed as the list of removed candidates of $k_0[2,7,8,13] || k_1[0,2,8,10] || k_8[0,7,10,13] || u_7[0]$. For each guess of $k_0[0,5,10,15]$, there exist 2^{88} such candidates, and the list is refreshed for each guess, hence, we only need 2^{88} 104-bit block of memory for step 7. Therefore, the overall memory complexity is about $2^{101.64}$ 128-bit memory blocks.

7.5 Summary

In this chapter, we have studied the security of the tweakable block cipher Kiasu-BC with respect to impossible differential cryptanalysis. In contrast to the designers' claim, we have presented an impossible differential attack on 8-round Kiasu-BC by adopting one of the existing impossible differential distinguishers on AES-128 and exploiting the tweak to have a zero difference round and hence extend the differential path by one additional analysis round in comparison to the impossible differential attacks on AES-128.

Chapter 8

Impossible Differential Properties of Reduced Round Streebog

In this chapter, we investigate the impossible differential properties of the underlying block cipher and compression function of the cryptographic hashing standard of the Russian federation Streebog. Our differential trail is constructed in such a way that allows us to recover the key of the underlying block cipher by observing input and output pairs of the compression function which utilizes the block cipher in Miyaguchi-Preneel mode. We discuss the implication of this attack when utilizing Streebog to construct a MAC using the secret-IV construction.

Parts of this chapter have been published in [2].

8.1 Introduction

In late 2012, Streebog [138] was announced as the Russian cryptographic hashing standard GOST R 34.11-2012. It officially replaced GOST R 34.11-94 which has been theoretically broken in [155] and further analyzed in [109, 110]. The output length of the Streebog hash function can be either 512 or 256-bit. Its compression function is based on a 12-round AES-like block cipher with 8×8-byte internal state, followed by an XOR operation with a whitening

key. The compression function operates in Miyaguchi-Preneel mode and is plugged in Merkle-Damgård domain extender with a finalization step [84].

Literature related to the cryptanalysis of Streebog includes the analysis of the collision resistance of its compression function and internal cipher by AlTawy et al. [7], and Wang et al. [151]. An integral analysis of the compression function has been presented by AlTawy and Youssef where integral distinguishers for the reduced compression function was proposed [8]. Moreover, preimage attacks on the reduced hash function have been independently proposed by AlTawy and Youssef [9], and Zou et al. [155], and later the attacks were improved by Bingka et al. [105]. In addition, Kazymyrov and Kazymyrova presented an analysis of the algebraic aspects of the function [84], and a long second preimage attack was proposed by Guo et al. [70]. Finally, a malicious version of the whole hash function was presented in [12], and a differential fault analysis when the function is used in different MAC schemes was proposed [10].

As explained in section 1.2, a MAC scheme [17] is a symmetric-key construction that provides mutual entity authentication and data integrity. Two common approaches are used to construct MAC schemes. The first approach employs a block cipher or a permutation, e.g., Cipher Block Chaining (CBC)-MAC [76], PELICAN-MAC [55], and ALPHA-MAC [54]. The second approach is based on hash functions where a secret key shared between the communicating parties is processed in a specific construction by the hash function, which is consequently viewed as a keyed hash function. Examples of this approach include simple prefix MAC [124], secret-IV MAC [124], NMAC [17], and the internationally standardized HMAC [17]. Attacks on MAC schemes usually aim to investigate their resistance against forgery attacks and key recovery attacks. The latter attack is more devastating since it directly grants the attacker the ability to impersonate any of the communicating parties and consequently forge any given message. As a result, analyzing hash-based MACs with respect to key recovery attacks has been the main aspect of many proposed works [68, 72].

When considering a hash function in a given MAC scheme, the first step is to analyze the security of the underlying primitives operating in the secret key model against key recovery attacks. Consequently, key recovery attacks on the underlying primitives has been considered as a valuable analytic model for the hash function. Such model has been adopted by Bouillaguet et al. in their analysis of the SHA-3 submission Lesamnta [38], where they presented a key recovery attack on the internal cipher reduced to 22 rounds. Additionally, the cryptanalysis of the SHA-3 submission EDON-R [98], where Laurent presented a key recovery attack on the function when used in the Secret-IV MAC.

One of the prospective applications of Streebog, as any other hash function, is using it in MAC schemes. Though both the simple prefix and the secret-IV MACs are vulnerable to length extension attacks, and the nested HMAC construction is internationally standardized, Streebog is by design not vulnerable to length extension attacks. This property may tempt users to adopt simpler MAC constructions such as the secret-IV setting. In this approach, the standard initial value is replaced by the secret key in the iterative construction of the hash function. More formally, $MAC(M) = H(K, M)$, where $H(K, M)$ is the hash value of the message M using the secret key K as the IV. Indeed, the designers of the NIST SHA-3 hash function, keccak [18,43], state on their website that since keccak is not vulnerable to length extension attacks, it does not need HMAC and propose that MAC computation can be done by concatenating the key with the message [85]. It should also be noted that the proof of security of the Miyaguchi-Preneel mode assumes that the underlying block cipher is ideal and must exhibit no distinguishing property. Accordingly, the results presented in this chapter are also interesting from this perspective since they are relevant to these indistinguishability claims.

In this chapter, we provide a security evaluation of Streebog's compression function when used in the secret key model where the IV is replaced by a secret key. More precisely, we present an Impossible Differential property of the underlying block cipher and compression function and employ it to recover the secret-IV of the compression function. Table 8.1 provides

a summary of current cryptanalytic results on the Streebog hash function.

Target	#Rounds	Time	Memory	Data	Attack	Reference
Internal cipher	5	2^8	2^8	-	Free-start collision	[7]
	8	2^{64}	2^8	-		
	3.75	-	-	-	ID distinguisher	Sec. 3
Internal permutation	6.5	2^{64}	-	2^{64}	Distinguisher	[8]
	7.5	2^{120}	-	2^{120}		
Compression function	7.75	2^{184}	2^8	-	Semi free-start collision	[7]
	4.75	2^8	-	-		
	7.75	2^{72}	2^8	-	Semi free-start near collision	
	8.75	2^{128}	2^8	-		
	9.75	2^{184}	2^8	-		
	6.75	$2^{399.5}$	2^{349}	$2^{483.1}$	Secret-IV recovery	Sec. 4
	6	2^{64}	-	2^{64}	Distinguisher	[8]
	7	2^{120}	-	2^{120}		
Hash function	5	2^{122}	2^{64}	-	Collision	[155]
	6	2^{496}	2^{64}	-	Preimage	[105]
	12	-	2^{14}	-	Differential fault analysis	[10]
	12	2^{266}	–	-	Long second preimage	[70]

Table 8.1: Summary of the current cryptanalytic results on Streebog.

The rest of this chapter is organized as follows. In section 8.2, the specification of the Streebog hash function along with the notation used throughout the chapter are provided. A brief recall of impossible differential cryptanalysis and how it can be applied to the compression function of Streebog is given in Section 8.3. Afterwards, in Section 8.4, we provide detailed description of the impossible differential attack on the underlying block cipher and the complexity of the attack. Finally, this chapter is summarized in Section 8.5.

8.2 Specification of Streebog

Streebog outputs a 512 or 256-bit hash value and can process up to 2^{512}-bit message. The compression function iterates over 12 rounds of an AES-like cipher with an 8×8 byte internal state and a final round of key mixing. The compression function operates in Miyaguchi-Preneel mode and is plugged in Merkle-Damgård domain extender with a finalization step.

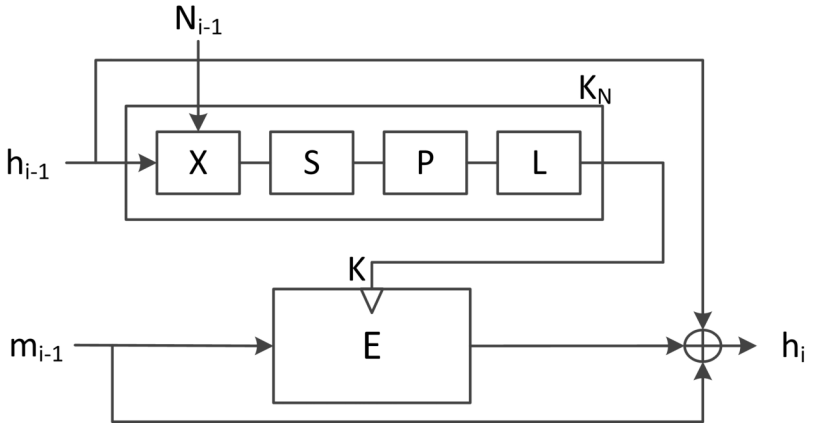

Figure 8.1: Streebog's compression function g_N

The input message M is padded into a multiple of 512 bits by appending one followed by zeros. Given $M = m_n \| .. \| m_1 \| m_0$, the compression function g_N is fed with three inputs: a chaining value h_{i-1}, a message block m_{i-1}, and a block size counter $N_{i-1} = 512 \times i$. (see Figure 8.1). Let h_i be a 512-bit chaining variable. The first state is loaded with the initial value IV and assigned to h_0. The hash value of M is computed as follows:

$$h_i \leftarrow g_N(h_{i-1}, m_{i-1}, N_{i-1}) \text{ for } i = 1, 2, .., n+1$$
$$h_{n+2} \leftarrow g_0(h_{n+1}, |M|, 0)$$
$$h(M) \leftarrow g_0(h_{n+2}, \sum(m_0, .., m_n), 0),$$

where $h(M)$ is the hash value of M. As depicted in Figure 8.1, the compression function g_N consists of:

- K_N: a nonlinear whitening round of the chaining value. It takes a 512-bit chaining vari-

able h_{i-1} and a block size counter N_{i-1} and outputs a 512-bit key K.

- E: an AES-based cipher that iterates over the message for 12 rounds in addition to a finalization key mixing round. The cipher E takes a 512-bit key K and a 512-bit message block m as a plaintext. As shown in Figure 8.2, it consists of two similar parallel flows for the state update and the key scheduling.

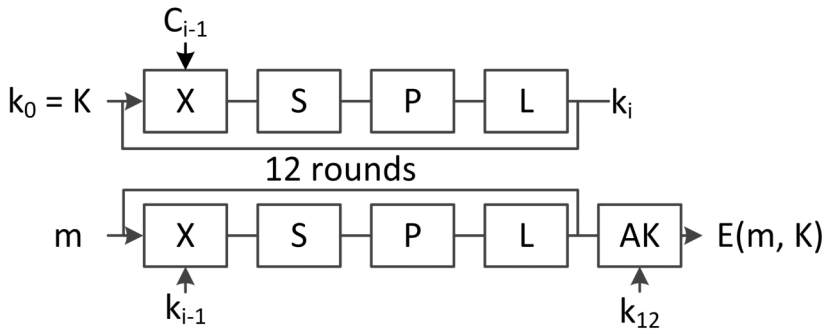

Figure 8.2: The internal block cipher (E)

Both K_N and E operate on an 8×8 byte key state K. E updates an additional 8×8 byte message state M. In one round, the state is updated by the following sequence of transformations

- AddKey(X): XOR with either a round key, a constant, or a block size counter (N)
- SubBytes (S): A nonlinear byte bijective mapping.
- Transposition (P): Byte permutation.
- LinearTransformation (L): Left multiplication by an MDS matrix in GF(2).

Initially, the key state K is loaded with the chaining value h_{i-1} and updated by K_N as follows:

$$k_0 = L \circ P \circ S \circ X(N_{i-1})$$

Now K contains the key k_0 to be used by the cipher E. The message state M is initially loaded with the message block m and $E(k_0, m)$ runs the key scheduling function on state K to generate 12 round keys $k_1, k_2, .., k_{12}$ as follows:

$$k_i = L \circ P \circ S \circ X(C_{i-1}), \text{ for } i = 1, 2, .., 12,$$

where C_{i-1} is the i^{th} round constant. The state M is updated as follows:

$$M_i = L \circ P \circ S \circ X(k_{i-1}), \text{ for } i = 1, 2, ..., 12.$$

The final round output is given by $E(k_0, m) = M_{12} \oplus k_{12}$. The output of g_N in the Miyaguchi-Preneel mode is $E(K_N(h_{i-1}, N_{i-1}), m_{i-1}) \oplus m_{i-1} \oplus h_{i-1}$. For further details, the reader is referred to [138].

8.2.1 Notation

Let M be an (8×8)-byte state denoting an input message state. The following notations are used throughout this chapter:

- M_i^I: A state at the beginning of round i.
- M_i^X, M_i^S, M_i^P and M_i^O: The message state at round i after the application of AddKey, SubBytes, Transposition and Linear Transformation, respectively. intuitively, M_{i-1}^O = M_i^I for $i >= 2$.
- $M_i[r, c]$: A byte at row r and column c of state M_i. Another representation of state bytes is an enumeration $0, 1, 2, 3, ..., 63$ as shown in Figure 8.3.
- M_i[row r]: Eight bytes located at row r of M_i state.
- M_i[col c]: Eight bytes located at column c of M_i state.

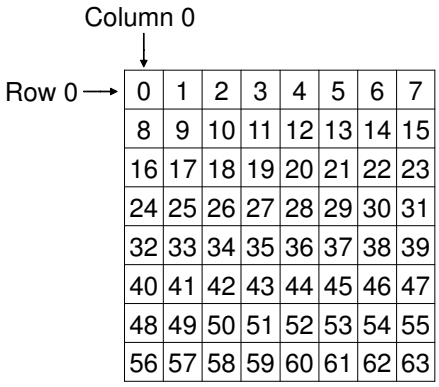

Figure 8.3: The 8×8 state of Streebog

8.3 Impossible Differential Cryptanalysis of the Compression Function

Although Streebog's compression function employs an AES-like cipher, applying the commonly used 4-round impossible differential of AES as is on Streebog's compression function would not be of value. In this case, we would recover the key of the last round of the block cipher masked by the chaining value (recall that Streebog's compression function works in Miyaguchi-Preneel mode). Therefore, we opted to reverse the impossible differential as detailed below to help recover the key of the first round, i.e., k_0. Since the key scheduling is bijective, once k_0 is recovered, we can recover the secret chaining value in the case of a secret-IV MAC construction when applied only at the level of the compression function. We note that the impossible differential property of Streebog's compression function would be limited to 3.75-rounds because, unlike AES, in the Streebog underlying block cipher, the linear transformation in the last round is not omitted.

As depicted in Figure 8.4, this impossible differential property states that given a pair of M_i^I with any 7 active bytes in the same arbitrary row (row 0 is chosen in the figure for illustration purposes), M_{i+3}^P cannot have only one active byte (similarly, that active byte can be any byte out of the 64-byte state). The deterministic differentials in this property are different from those

of the AES property; on top of being swapped, the forward differential is just 1-round while the backward differential is 2.75 rounds. The property is rationalized as follows: any 7 active bytes in the same row of M_i^I give 56 active bytes by M_i^O with one entire row being equal (the position of the zero-difference byte in the input will determine the position of the row). On the other hand, one active byte in M_{i+3}^P leads to a full active state where all 64 bytes are active in M_{i+1}^I, which means that the middle states contradict with each other as illustrated in Figure 8.4. As explained above, this impossible differential holds regardless of the row and the positions within that row.

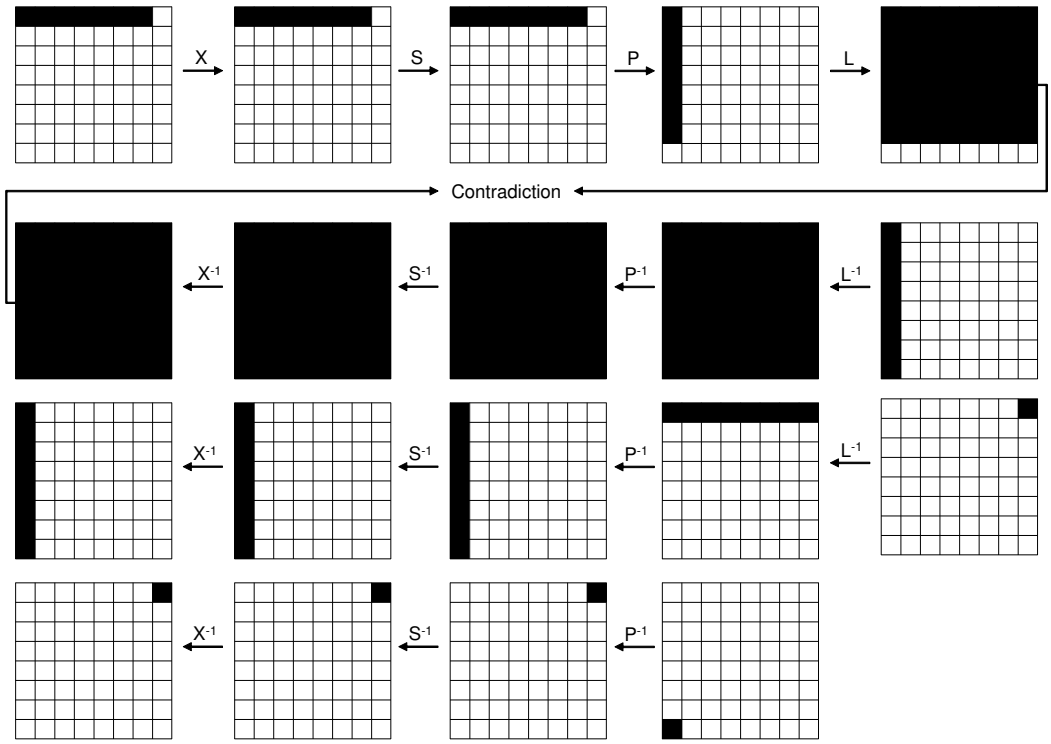

Figure 8.4: Impossible differential property of the internal block cipher

Figure 8.5 gives an example of impossible input and output patterns for the compression function. When the compression function message input has specific non-zero difference at bytes 0 to 6 (δ_0 to δ_6 in the figure) and zero difference in all the other bytes, then after the feedforward the output difference cannot have the same values as the input difference at bytes 0 to 6 (δ_0 to δ_6) and a non-zero difference at byte 56 (δ_7). Such input and output difference

patterns are impossible on the compression function level. It is to be noted that there exists $8 \times 8 \times 255^7$ such input patterns (the input differences can be in any row and the inactive byte can be at any column of that row and each of the differences can take $2^8 - 1$ possible values. There are also $((7 \times 8) + 1) \times 255 = 57 \times 255$ contradicting output patterns (the non-zero output difference byte can be at any column of 7 rows, i.e., all but the input differences row and takes only the position of the inactive byte on the input differences row).

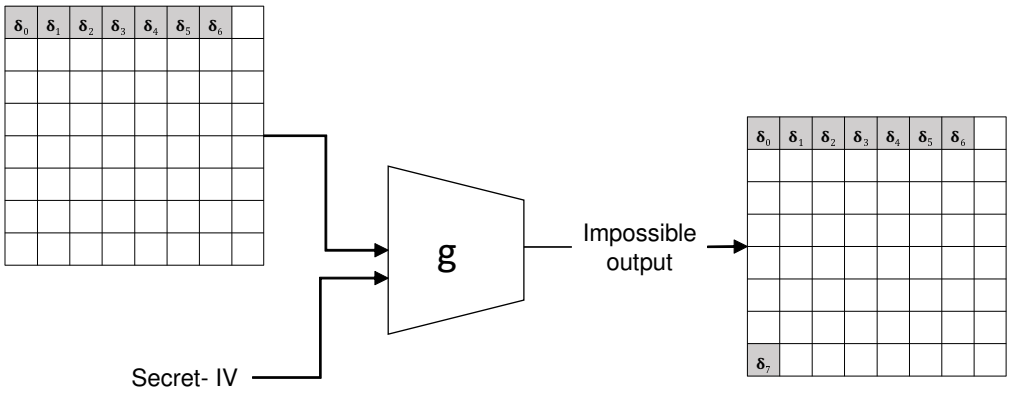

Figure 8.5: Example of impossible differentials for the 3.75 round reduced compression function

8.4 Impossible Differential Attack on 6.75 rounds of Streebog's Compression Function

Considering the aforementioned impossible differential property, a 6.75-rounds attack on Streebog's compression function can be mounted as detailed hereafter. In our attack model, we assume access to the keyed Streebog's reduced compression function oracle which allows us to query the keyed oracle with chosen messages and get the corresponding compression function outputs.

The attack is illustrated in Figure 8.6 and proceeds as follows:

1. The keyed compression function oracle is fed with 2^n structures where each structure

consists of 2^{256} messages having the same value in columns 4, 5, 6 and 7 and assuming all possible 2^{256} values in columns 0, 1, 2 and 3. Accordingly, each structure offers $2^{256} \times 2^{256} \times 1/2 \simeq 2^{511}$ pairs of messages. Thus, we have a total of 2^{n+256} messages, and 2^{n+511} message pairs for the 2^n structures.

2. As we have access to the output of the compression function, which operates in Miyaguchi-Preneel mode as depicted in Figure 8.1, the output h_i that we observe is $h_{i-1} \oplus m_{i-1} \oplus c_{i-1}$ where c_{i-1} is the corresponding output of the reduced variant of the block cipher when its input is m_{i-1}. Therefore, for each message query, we first XOR the compression function output with the corresponding input message, i.e., $h_i \oplus m_{i-1}$, to get $c_{i-1} \oplus h_{i-1}$ and keep only the pairs that have non-zero difference in just one column. Consequently, it is expected to have $2^{n+511} \times 2^{-448} = 2^{n+63}$ pairs.

3. Next, we randomly assume a 256-bit value of the first round key at columns 0, 1, 2 and 3, i.e., $k_0[\text{col } 0, 1, 2, 3]$, partially encrypt these 4 columns of the message pairs corresponding to the remaining ciphertext pairs, i.e., we compute $M_1^O[\text{row } 0, 1, 2, 3] = L \circ P \circ S[M_1^I[\text{col } 0, 1, 2, 3] \oplus k_0[\text{col } 0, 1, 2, 3]]$ and we choose the pairs which have just one non-zero difference byte at column 0 of the 4 rows, as shown in Figure 8.6. The probability of such combination is $q_1 = (2^{-8})^7 \times (2^{-8})^7 \times (2^{-8})^7 \times (2^{-8})^7 = 2^{-224}$. Accordingly, $2^{n+63} \times 2^{-224} = 2^{n-161}$ message pairs are expected to pass this step.

4. Afterwards, we assume a 32-bit value for the bytes 0, 1, 2 and 3 of column 0 of the key k_1, i.e., bytes 0, 8, 16, 24 as in Figure 8.3, partially encrypt these bytes through the second round to compute $M_2^O[\text{row } 0] = L \circ P \circ S[M_2^I[(0, 1, 2, 3), 0] \oplus k_1[(0, 1, 2, 3), 0]$. We choose the pairs that have only one zero-difference byte at any column of that row. The probability of such pairs is $q_2 = 8 \times 2^{-8} = 2^{-5}$. So after this step, we have $2^{n-161} \times 2^{-5} = 2^{n-166}$ message pairs.

5. Then, we assume a 64-bit value for the last column of $k_7[\text{col } 7] \oplus h_{i-1}[\text{col } 7]$ so that we end up with the block cipher output (Note: h_{i-1} is the targeted secret-IV to be recovered and

it has the same value for all the pairs we have). Specifically, we calculate $M_7^P[\text{col } 7] = (c_{i-1}[\text{col } 7] \oplus h_{i-1}[\text{col } 7]) \oplus (k_7[\text{col } 7] \oplus h_{i-1}[\text{col } 7])$ for all the pairs we have so far. The former value is the output we get from step 2, while the latter is the value we just assumed.

6. For each of the filtered pairs, we partially decrypt column 7. In other words, we compute $L^{-1} \circ ((S^{-1} \circ P^{-1}(M_7^P[\text{col } 7])) \oplus (S^{-1} \circ P^{-1}(M_7^{*P}[\text{col } 7]))$ and we choose the pairs which have only one non-zero difference byte at any position of row 7, which happens with probability $p = 8 \times (2^{-8})^7 = 2^{-53}$. This difference is impossible, hence each key (or to be exact each $k_7 \oplus h_{i-1}$ i.e., $k_7 \oplus const$) that results in such a difference is a wrong key. Therefore, after analyzing 2^{n-166} pairs, only $2^{64} \times (1 - 2^{-53})^{2^{n-166}} \simeq 2^{64} \times (e^{-1})^{2^{n-219}} \simeq 2^{64} \times 2^{-1.4 \times (2^{n-219})}$ wrong values of the last column of $k_7 \oplus h_{i-1}$ remains.

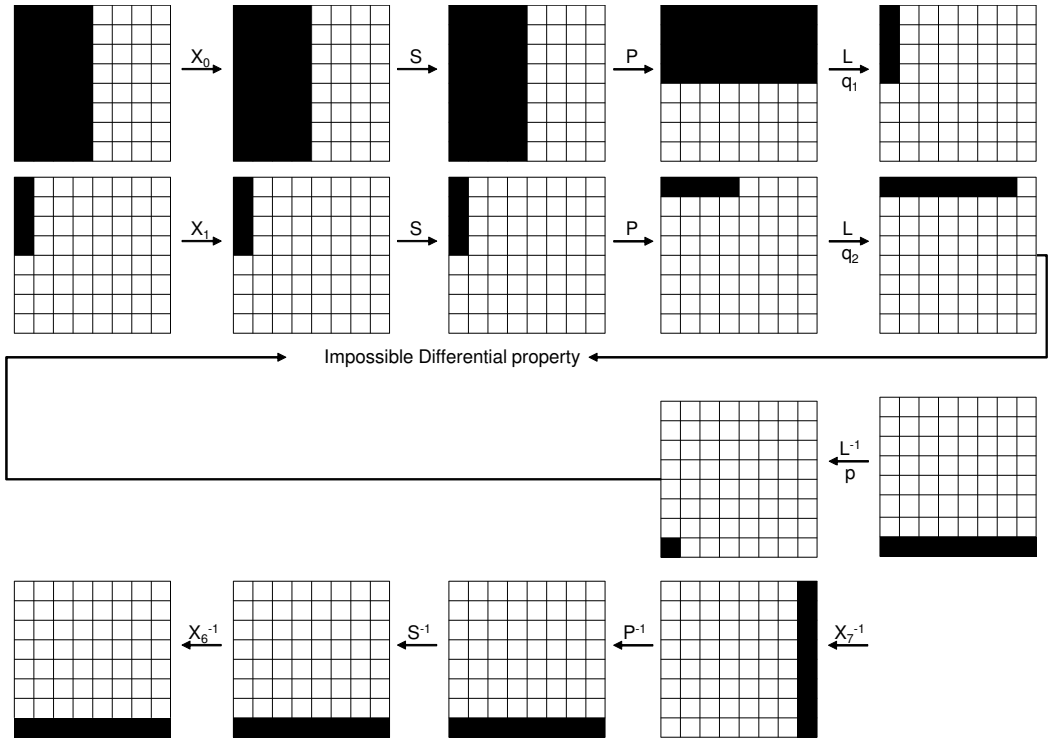

Figure 8.6: 6.75-rounds impossible differential attack on the compression function

To be able to find the correct partial keys, we discard the 64-bit values for $k_7 \oplus h_{i-1}$ unless the initial guess of the 256-bit value of k_0 and the 32-bit value of k_1 is correct. The wrong values (k_0, k_1, k_7) remain with probability: $2^{(256+32)} \times 2^{64} \times 2^{-1.4 \times (2^{n-219})} = 2^{352-1.4 \times (2^{n-219})}$ which

should be made as small as possible, e.g., less than 2^{-30} (that value is chosen to maximize the probability of finding the correct tuple without having a significant impact on the number of messages needed) which means $2^{352-1.4\times(2^{n-219})} < 2^{-30}$ resulting in $n > 227.09$. Accordingly, when we start with $2^{227.1}$ structures and there remains a value of $k_7 \oplus h_{i-1}$, we consider the assumed 256-bit value for k_0 is correct and the probability of wrong value of (k_0, k_1, k_7) is $2^{-32.1}$.

Attack Complexity. With n set to 227.1, the attack requires $2^{n+256} = 2^{483.1}$ chosen messages. The time complexity of the attack is calculated as follows:

- In step 3, row 0 requires $2 \times 2^{64} \times 2^{n+63} \times 1/8 = 2^{n+125}$ one round encryptions, row 1 requires $2 \times 2^{64} \times 2^{64} \times 2^{n+7} \times 1/8 = 2^{n+133}$ one round encryptions, row 2 requires $2 \times 2^{64} \times 2^{64} \times 2^{64} \times 2^{n-49} \times 1/8 = 2^{n+141}$ one round encryptions and row 3 requires $2 \times 2^{64} \times 2^{64} \times 2^{64} \times 2^{64} \times 2^{n-105} \times 1/8 = 2^{n+149}$ one round encryptions.

- In step 4, row 0 requires $2 \times 2^{256} \times 2^{32} \times 2^{n-161} \times 1/16 = 2^{n+124}$ one round encryptions.

- In step 6, column 7 decryption requires $2 \times 2^{256} \times 2^{32} \times 2^{64} \times (1 + (1 - 2^{-53}) + (1 - 2^{-53})^2 + ... + (1 - 2^{-53})^{2^{n-166}}) \times 1/16 \simeq 2^{402}$ one round encryptions.

- For $n = 227.1$, the overall complexity of the attack is about $(2^{352.1} + 2^{360.1} + 2^{368.1} + 2^{376.1} + 2^{351.1} + 2^{402})/6.75 \simeq 2^{399.5}$ encryptions to recover 256 bits of k_0.

Then, the other half of k_0 can be found by an exhaustive search. Hence, the whole k_0 can be recovered with time complexity of $2^{399.5} + 2^{256} \simeq 2^{399.5}$ queries. Once k_0 is recovered, we can easily recover the secret-IV h_{i-1}. Finally, the memory requirement is dominated by the memory needed to store the list of the deleted key tuples (k_0,k_1,k_7), so we need $2^{352}/2^3 = 2^{349}$ bytes.

8.5 Summary

In this chapter, we have analyzed Streebog's compression function in the secret key model. More precisely, we have proposed Secret-IV recovery attack, at the level of the compression function, based on impossible differential properties of the compression function. The attack requires $2^{483.1}$ messages, has a time complexity equivalent to $2^{399.5}$ queries to the compression function reduced to 6.75 rounds and needs 2^{349} bytes of memory. Finally, it should be noted that this attack does not directly contradict the security claims of Streebog and does not present any immediate practical threat to its security. However, it helps as a cautionary note for using Streebog in this mode since it might be tempting to do so because the finalization stage of Streebog is strengthened against length extension attacks that are the main reasons for not using secret-IV or secret-prefix MAC constructions.

Chapter 9

Summary and Future Research Directions

9.1 Summary of contributions

In this section, we briefly summarize the accomplished work and the major contributions of this thesis. In chapter 1, we presented the motivation for this work and in chapter 2, we provided the essential background by formally introducing block ciphers and their most common cryptanalysis techniques.

In chapter 3, we addressed the limitation of the existing MILP modeling approaches and proposed an efficient way to generate the linear inequalities representing the differential propagation through large S-boxes (6-bit and upwards). Our proposed approach starts by separating the DDT entries of the S-box by their values where a Boolean function is assigned to each distinct value. Then, each Boolean function is converted to its minimized product-of-sum form that can be efficiently represented by a set of linear inequalities. Our approach allowed us to precisely evaluate the probability of differential characteristics. It has been applied to two of the efficient AES-round function based constructions proposed in FSE 2016 by Jean and Nikolić. It helped us improve the lower bound on the number of the active S-boxes in one construction and the upper bound on the differential characteristic of the other. Although our approach was applied in the context of differential cryptanalysis, it can easily be applied in other techniques

such as linear cryptanalysis.

Then, in chapter 4, we analyzed the security of the Japanese Hierocrypt-L1 block cipher against MitM attack in the single-key setting. In particular, we constructed a five S-box layer distinguisher that we utilized to launch a meet-in-the-middle attack on 8 S-box layer round-reduced Hierocrypt-L1 using the differential enumeration technique. Our attack allowed us to recover the master key with data complexity of 2^{49} chosen plaintexts, time complexity of $2^{114.8}$ 8 S-box layers Hierocrypt-L1 encryptions and memory complexity of 2^{106} 64-bit blocks. This was the first cryptanalysis result that reaches eight S-box layers of Hierocrypt-L1 in the single-key setting.

In chapter 5, we presented two meet-in-the-middle attacks in the single-key setting on eight S-box layer reduced version of another Japanese block cipher, i.e., Hierocrypt-3 with 256-bit key. The first attack was based on the differential enumeration approach where we proposed a truncated differential characteristic in the first six S-box layers and matched a multiset of state differences at its output. The other attack was based on the original meet-in-the-middle attack strategy proposed by Demirci and Selçuk at FSE 2008 to attack reduced versions of both AES-192 and AES-256. The master key was recovered with data complexity of 2^{113} chosen plaintexts, time complexity of 2^{238} eight S-box layer reduced Hierocrypt-3 encryptions and memory complexity of 2^{218} 128-bit blocks in the attack based on the differential enumeration. The data, time and memory complexities of the second attack were 2^{32}, 2^{245} and 2^{239}, respectively. Again, these were the first attacks on eight S-box layer reduced Hierocrypt-3.

In chapter 6, we presented an impossible differential attack on the ARX-based block cipher SPARX-64/128. Particularly, we presented 12 and 13-round impossible distinguishers on SPARX-64/128 that can be used to attack 15 and 16-round with post-whitening keys, respectively. While the 15-round attack started from round zero, the 16-round one, exploiting the key schedule, had to start from round two.

Continuing with the impossible differential attack, in chapter 7, we presented an impossible differential attack on Kiasu-BC that is one of the instantiations of the TWEAKEY framework. It can be considered as tweakable AES-128. Contrary to the designers' claim, we proved that the security of Kiasu-BC is not identical to the security of AES-128 when it comes to impossible differential attacks. More precisely, we showed that an 8-round impossible differential attack on Kiasu-BC is feasible by adopting one of the existing impossible differential distinguishers on AES-128 and exploiting the tweak to gain this extra round in comparison to the impossible differential attacks on AES-128.

Lastly, in chapter 8, we investigated the impossible differential properties of the underlying block cipher and compression function of the cryptographic hashing standard of the Russian federation Streebog. Our differential trail was constructed in such a way that allowed us to recover the key of the underlying block cipher by observing input and output pairs of the compression function that utilizes the block cipher in Miyaguchi-Preneel mode. We discussed the implication of this attack when utilizing Streebog to construct a MAC scheme using the secret-IV construction and pointed out that although the finalization stage of Streebog is strengthened against length extension attacks, using it in the secret-IV mode might incur some risks of exposing the secret key.

9.2 Future work

Here below, we list some topics that would be of interest for future research.

- We have applied our new proposed MILP modeling to represent the differential propagation through an 8-bit S-box to two AES-round based constructions. Hence, it would be of interest to find other meaningful applications with maybe less uniform S-boxes. Moreover, we have emphasized the applicability of our approach in other cryptanalysis techniques so it would be equally interesting to use it in linear cryptanalysis, impossible

differential cryptanalysis or zero-correlation linear cryptanalysis.

- We have studied the security of some symmetric-key primitives under the classical black-box attack model, which assumes that the parties of the communication are trusted. It also assumes that the primitive is always executed in a secure environment. This implies that the capabilities of the adversary are limited to having access to the inputs and outputs of such a symmetric-key primitive. With the proliferation of computer and communication devices and the wide spread of the Internet and mobile networks, other equally important attack models emerged, e.g., the gray-box and the white-box attack models. In the gray-box attack model, implementation-specific information, such as the execution time or the power consumption, is exploited as such information is typically correlated to the secret key. The white-box attack model assumes an even more powerful adversary who has full access to the software implementation of a symmetric-key primitive as well as full control over its execution environment. This means that this powerful attacker can use debuggers with breakpoints to observe and change, if desired, intermediate results of the software implementation. Therefore, it would be of great interest to analyze symmetric-key primitives under these other attack models, especially, the white-box one that has not been well studied in academia.

- We have analyzed the impossible differential properties of the underlying block cipher of the Streebog hash function and extended it to its compression function. The results in [64] show that the modular sum finalization stage weakens the function when used in HMAC construction, so it would be of interest to extend our attack to the full hash function as it requires further investigation of the compression function one wayness properties.

Bibliography

[1] A. Abdelkhalek, R. AlTawy, M. Tolba, and A. M. Youssef. Meet-in-the-Middle Attacks on Reduced-Round Hierocrypt-3. In K. E. Lauter and F. Rodríguez-Henríquez, editors, *Progress in Cryptology - LATINCRYPT 2015 - 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings*, volume 9230 of *Lecture Notes in Computer Science*, pages 187–203. Springer, 2015. ISBN 978-3-319-22173-1.

[2] A. Abdelkhalek, R. AlTawy, and A. M. Youssef. Impossible Differential Properties of Reduced Round Streebog. In S. E. Hajji, A. Nitaj, C. Carlet, and E. M. Souidi, editors, *Codes, Cryptology, and Information Security - First International Conference, C2SI 2015, Rabat, Morocco, May 26-28, 2015, Proceedings - In Honor of Thierry Berger*, volume 9084 of *Lecture Notes in Computer Science*, pages 274–286. Springer, 2015. ISBN 978-3-319-18680-1.

[3] A. Abdelkhalek, Y. Sasaki, Y. Todo, M. Tolba, and A. M. Youusef. MILP Modeling for (Large) S-boxes to Optimize Probability of Differential Characteristics. *IACR Trans. Symmetric Cryptol.*, 2017(4), 2017. To appear.

[4] A. Abdelkhalek, M. Tolba, and A. M. Youssef. Improved Key Recovery Attack on Round-reduced Hierocrypt-L1 in the Single-Key Setting. In R. S. Chakraborty, P. Schwabe, and J. A. Solworth, editors, *Security, Privacy, and Applied Cryptography Engineering - 5th International Conference, SPACE 2015, Jaipur, India, October 3-7,*

2015, Proceedings, volume 9354 of *Lecture Notes in Computer Science*, pages 139–150. Springer, 2015. ISBN 978-3-319-24125-8.

[5] A. Abdelkhalek, M. Tolba, and A. M. Youssef. Impossible Differential Attack on Reduced Round SPARX-64/128. In M. Joye and A. Nitaj, editors, *Progress in Cryptology - AFRICACRYPT 2017 - 9th International Conference on Cryptology in Africa, Dakar, Senegal, May 24-26, 2017, Proceedings*, volume 10239 of *Lecture Notes in Computer Science*, pages 135–146, 2017. ISBN 978-3-319-57338-0.

[6] R. AlTawy, A. Abdelkhalek, and A. M. Youssef. A Meet-in-the-Middle Attack on Reduced-Round Kalyna-*b*/2*b*. *IEICE Transactions*, 99-D(4):1246–1250, 2016.

[7] R. AlTawy, A. Kircanski, and A. M. Youssef. Rebound Attacks on Stribog. In H. Lee and D. Han, editors, *Information Security and Cryptology - ICISC 2013 - 16th International Conference, Seoul, Korea, November 27-29, 2013, Revised Selected Papers*, volume 8565 of *Lecture Notes in Computer Science*, pages 175–188. Springer, 2013. ISBN 978-3-319-12159-8.

[8] R. AlTawy and A. M. Youssef. Integral Distinguishers for Reduced-round Stribog. *Inf. Process. Lett.*, 114(8):426–431, 2014.

[9] R. AlTawy and A. M. Youssef. Preimage Attacks on Reduced-Round Stribog. In D. Pointcheval and D. Vergnaud, editors, *Progress in Cryptology - AFRICACRYPT 2014 - 7th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 28-30, 2014. Proceedings*, volume 8469 of *Lecture Notes in Computer Science*, pages 109–125. Springer, 2014. ISBN 978-3-319-06733-9.

[10] R. AlTawy and A. M. Youssef. Differential Fault Analysis of Streebog. In J. Lopez and Y. Wu, editors, *Information Security Practice and Experience - 11th International Conference, ISPEC 2015, Beijing, China, May 5-8, 2015. Proceedings*, volume 9065

of *Lecture Notes in Computer Science*, pages 35–49. Springer, 2015. ISBN 978-3-319-17532-4.

[11] R. AlTawy and A. M. Youssef. Meet in the Middle Attacks on Reduced Round Kuznyechik. Cryptology ePrint Archive, Report 2015/096, 2015. http://eprint.iacr.org/2015/096.

[12] R. AlTawy and A. M. Youssef. Watch your Constants: Malicious Streebog. *IET Information Security*, 9(6):328–333, 2015.

[13] B. Bahrak and M. R. Aref. Impossible differential attack on seven-round AES-128. *IET Information Security*, 2(2):28–32, 2008.

[14] P. S. L. M. Barreto, V. Rijmen, J. N. Jr., B. Preneel, J. Vandewalle, and H. Y. Kim. Improved SQUARE Attacks against Reduced-Round HIEROCRYPT. In M. Matsui, editor, *Fast Software Encryption, 8th International Workshop, FSE 2001 Yokohama, Japan, April 2-4, 2001, Revised Papers*, volume 2355 of *Lecture Notes in Computer Science*, pages 165–173. Springer, 2001. ISBN 3-540-43869-6.

[15] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404, 2013. http://eprint.iacr.org/2013/404.

[16] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. SIMON and SPECK: Block Ciphers for the Internet of Things. Cryptology ePrint Archive, Report 2015/585, 2015. http://eprint.iacr.org/2015/585.

[17] M. Bellare, R. Canetti, and H. Krawczyk. Keying Hash Functions for Message Authentication. In N. Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1996. ISBN 3-540-61512-1.

[18] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Keccak sponge function family main document, 2009.

[19] E. Biham. New Types of Cryptanalytic Attacks Using Related Keys. *J. Cryptology*, 7(4): 229–246, 1994.

[20] E. Biham. How to decrypt or even substitute DES-encrypted messages in 2^{28} steps. *Inf. Process. Lett.*, 84(3):117–124, 2002.

[21] E. Biham, A. Biryukov, and A. Shamir. Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. *J. Cryptology*, 18(4):291–311, 2005.

[22] E. Biham, O. Dunkelman, and N. Keller. The Rectangle Attack - Rectangling the Serpent. In B. Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *Lecture Notes in Computer Science*, pages 340–357. Springer, 2001. ISBN 3-540-42070-3.

[23] E. Biham, O. Dunkelman, and N. Keller. Enhancing Differential-Linear Cryptanalysis. In Y. Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, volume 2501 of *Lecture Notes in Computer Science*, pages 254–266. Springer, 2002. ISBN 3-540-00171-9.

[24] E. Biham and N. Keller. Cryptanalysis of reduced variants of Rijndael. 3rd AES Conference, New York, USA, 2000.

[25] E. Biham and A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *J. Cryptology*, 4(1):3–72, 1991.

[26] A. Biryukov. Structural Cryptanalysis. In H. C. A. van Tilborg and S. Jajodia, editors,

Encyclopedia of Cryptography and Security, 2nd Ed., page 1266. Springer, 2011. ISBN 978-1-4419-5905-8.

[27] A. Biryukov and D. A. Wagner. Slide Attacks. In L. R. Knudsen, editor, *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999, Proceedings*, volume 1636 of *Lecture Notes in Computer Science*, pages 245–259. Springer, 1999. ISBN 3-540-66226-X.

[28] A. Biryukov and D. A. Wagner. Advanced Slide Attacks. In B. Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, pages 589–606. Springer, 2000. ISBN 3-540-67517-5.

[29] J. Bisschop. AIMMS - Optimization Modeling, 2017. https://download.aimms.com/aimms/download/manuals/AIMMS3_OM.pdf.

[30] J. Black. The Ideal-Cipher Model, Revisited: An Uninstantiable Blockcipher-Based Hash Function. In M. J. B. Robshaw, editor, *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers*, volume 4047 of *Lecture Notes in Computer Science*, pages 328–340. Springer, 2006. ISBN 3-540-36597-4.

[31] J. Black. Authenticated Encryption. In H. C. A. van Tilborg and S. Jajodia, editors, *Encyclopedia of Cryptography and Security, 2nd Ed.*, pages 52–61. Springer, 2011. ISBN 978-1-4419-5905-8.

[32] C. Blondeau, G. Leander, and K. Nyberg. Differential-Linear Cryptanalysis Revisited. *J. Cryptology*, 30(3):859–888, 2017.

[33] A. Bogdanov. *Analysis and Design of Block Cipher Constructions*. PhD thesis, Ruhr University Bochum, 2010.

[34] A. Bogdanov, C. Boura, V. Rijmen, M. Wang, L. Wen, and J. Zhao. Key Difference Invariant Bias in Block Ciphers. In K. Sako and P. Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 357–376. Springer, 2013. ISBN 978-3-642-42032-0.

[35] A. Bogdanov, D. Khovratovich, and C. Rechberger. Biclique Cryptanalysis of the Full AES. In D. H. Lee and X. Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 344–371. Springer, 2011. ISBN 978-3-642-25384-3.

[36] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007. ISBN 978-3-540-74734-5.

[37] A. Bogdanov and V. Rijmen. Linear Hulls with Correlation Zero and Linear Cryptanalysis of Block Ciphers. *Des. Codes Cryptography*, 70(3):369–383, 2014.

[38] C. Bouillaguet, O. Dunkelman, G. Leurent, and P. Fouque. Attacks on Hash Functions Based on Generalized Feistel: Application to Reduced-Round *Lesamnta* and *SHAvite-3*$_{512}$. In A. Biryukov, G. Gong, and D. R. Stinson, editors, *Selected Areas in Cryptography - 17th International Workshop, SAC 2010, Waterloo, Ontario, Canada, August 12-13, 2010, Revised Selected Papers*, volume 6544 of *Lecture Notes in Computer Science*, pages 18–35. Springer, 2010. ISBN 978-3-642-19573-0.

[39] CAESAR. Competition for Authenticated Encryption: Security, Applicability, and Robustness. http://competitions.cr.yp.to/caesar.html.

[40] C. D. Canniere, I. Dinur, and A. Shamir. New Generic Attacks Which are Faster Than Exhaustive Search. Presented at the Rump Session of FSE'09, 2009.

[41] A. Canteaut. Stream Cipher. In H. C. A. van Tilborg and S. Jajodia, editors, *Encyclopedia of Cryptography and Security, 2nd Ed.*, pages 1263–1265. Springer, 2011. ISBN 978-1-4419-5905-8.

[42] F. Chabaud and S. Vaudenay. Links Between Differential and Linear Cryptanalysis. In A. D. Santis, editor, *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 356–365. Springer, 1994. ISBN 3-540-60176-7.

[43] S.-j. Chang, R. Perlner, W. E. Burr, M. S. Turan, J. M. Kelsey, S. Paul, and L. E. Bassham. *Third-round report of the SHA-3 cryptographic hash algorithm competition*. Citeseer, 2012. http://nvlpubs.nist.gov/nistpubs/ir/2012/\NIST.IR.7896.pdf.

[44] N. Courtois. General Principles of Algebraic Attacks and New Design Criteria for Cipher Components. In H. Dobbertin, V. Rijmen, and A. Sowa, editors, *Advanced Encryption Standard - AES, 4th International Conference, AES 2004, Bonn, Germany, May 10-12, 2004, Revised Selected and Invited Papers*, volume 3373 of *Lecture Notes in Computer Science*, pages 67–83. Springer, 2004. ISBN 3-540-26557-0.

[45] CRYPTEC. e-Government Candidate Recommended Ciphers List, 2013. http://www.cryptrec.go.jp/english/method.html.

[46] CRYPTEC. e-Government Recommended Ciphers List, 2003. http://www.cryptrec.go.jp/english/images/cryptrec_01en.pdf.

[47] CRYPTEC. Specification on a Block Cipher: Hierocrypt-3, May 2002. http://www.cryptrec.go.jp/cryptrec_03_spec_cypherlist_files/PDF/08_02espec.pdf.

[48] CRYPTEC. Specification on a Block Cipher: Hierocrypt-L1, September 2001. http://www.cryptrec.go.jp/cryptrec_03_spec_cypherlist_files/PDF/04_02espec.pdf.

[49] T. Cui, K. Jia, K. Fu, S. Chen, and M. Wang. New Automatic Search Tool for Impossible Differentials and Zero-Correlation Linear Approximations. Cryptology ePrint Archive, Report 2016/689, 2016. http://eprint.iacr.org/2016/689.

[50] J. Daemen, L. R. Knudsen, and V. Rijmen. The Block Cipher Square. In E. Biham, editor, *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings*, volume 1267 of *Lecture Notes in Computer Science*, pages 149–165. Springer, 1997. ISBN 3-540-63247-6.

[51] J. Daemen and V. Rijmen. The Wide Trail Design Strategy. In B. Honary, editor, *Cryptography and Coding, 8th IMA International Conference, Cirencester, UK, December 17-19, 2001, Proceedings*, volume 2260 of *Lecture Notes in Computer Science*, pages 222–238. Springer, 2001. ISBN 3-540-43026-1.

[52] J. Daemen and V. Rijmen. AES and the Wide Trail Design Strategy. In L. R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 108–109. Springer, 2002. ISBN 3-540-43553-0.

[53] J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002. ISBN 3-540-42580-2.

[54] J. Daemen and V. Rijmen. A New MAC Construction ALRED and a Specific Instance ALPHA-MAC. In H. Gilbert and H. Handschuh, editors, *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*, volume 3557 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2005. ISBN 3-540-26541-4.

[55] J. Daemen and V. Rijmen. The Pelican MAC Function. Cryptology ePrint Archive, Report 2005/088, 2005. http://eprint.iacr.org/2005/088.

[56] H. Demirci and A. A. Selçuk. A Meet-in-the-Middle Attack on 8-Round AES. In K. Nyberg, editor, *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, volume 5086 of *Lecture Notes in Computer Science*, pages 116–126. Springer, 2008. ISBN 978-3-540-71038-7.

[57] H. Demirci, I. Taskin, M. Çoban, and A. Baysal. Improved Meet-in-the-Middle Attacks on AES. In B. K. Roy and N. Sendrier, editors, *Progress in Cryptology - INDOCRYPT 2009, 10th International Conference on Cryptology in India, New Delhi, India, December 13-16, 2009. Proceedings*, volume 5922 of *Lecture Notes in Computer Science*, pages 144–156. Springer, 2009. ISBN 978-3-642-10627-9.

[58] P. Derbez, P. Fouque, and J. Jean. Improved Key Recovery Attacks on Reduced-Round AES in the Single-Key Setting. In T. Johansson and P. Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 371–387. Springer, 2013. ISBN 978-3-642-38347-2.

[59] Data Encryption Standard: FIPS. National Bureau of Standards, U.S. Department of Commerce, 1977.

[60] T. S. Developers. *SageMath, the Sage Mathematics Software System (Version 7.6)*, 2017. http://www.sagemath.org.

[61] W. Diffie and M. E. Hellman. Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard. *IEEE Computer*, 10(6):74–84, 1977.

[62] D. Dinu, L. Perrin, A. Udovenko, V. Velichkov, J. Großschädl, and A. Biryukov. Design Strategies for ARX with Provable Bounds: SPARX and LAX. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, pages 484–513, 2016.

[63] D. Dinu, L. Perrin, A. Udovenko, V. Velichkov, J. Großschädl, and A. Biryukov. Design Strategies for ARX with Provable Bounds: SPARX and LAX (Full Version). Cryptology ePrint Archive, Report 2016/984, 2016. http://eprint.iacr.org/2016/984.

[64] I. Dinur and G. Leurent. Improved Generic Attacks against Hash-Based MACs and HAIFA. In J. A. Garay and R. Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 149–168. Springer, 2014. ISBN 978-3-662-44370-5.

[65] I. Dinur and A. Shamir. Cube Attacks on Tweakable Black Box Polynomials. In A. Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 278–299. Springer, 2009. ISBN 978-3-642-01000-2.

[66] C. Dobraunig, M. Eichlseder, and F. Mendel. Square Attack on 7-Round Kiasu-BC. In M. Manulis, A. Sadeghi, and S. Schneider, editors, *Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22,*

2016. Proceedings, volume 9696 of *Lecture Notes in Computer Science*, pages 500–517. Springer, 2016. ISBN 978-3-319-39554-8.

[67] O. Dunkelman, N. Keller, and A. Shamir. Improved Single-Key Attacks on 8-Round AES-192 and AES-256. In M. Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 158–176. Springer, 2010. ISBN 978-3-642-17372-1.

[68] P. Fouque, G. Leurent, and P. Q. Nguyen. Full Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5. In A. Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *Lecture Notes in Computer Science*, pages 13–30. Springer, 2007. ISBN 978-3-540-74142-8.

[69] K. Fu, M. Wang, Y. Guo, S. Sun, and L. Hu. MILP-Based Automatic Search Algorithms for Differential and Linear Trails for Speck. In T. Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 268–288. Springer, 2016. ISBN 978-3-662-52992-8.

[70] J. Guo, J. Jean, G. Leurent, T. Peyrin, and L. Wang. The Usage of Counter Revisited: Second-Preimage Attack on New Russian Standardized Hash Function. In A. Joux and A. M. Youssef, editors, *Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers*, volume 8781 of *Lecture Notes in Computer Science*, pages 195–211. Springer, 2014. ISBN 978-3-319-13050-7.

[71] I. Gurobi Optimization. Gurobi Optimizer Reference Manual, 2016.

[72] H. Handschuh and B. Preneel. Key-Recovery Attacks on Universal Hash Function Based

MAC Algorithms. In D. A. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 144–161. Springer, 2008. ISBN 978-3-540-85173-8.

[73] M. E. Hellman. A Cryptanalytic Time-Memory Trade-off. *IEEE Trans. Information Theory*, 26(4):401–406, 1980.

[74] M. Hermelin, J. Y. Cho, and K. Nyberg. Multidimensional Extension of Matsui's Algorithm 2. In O. Dunkelman, editor, *Fast Software Encryption, 16th International Workshop, FSE 2009, Leuven, Belgium, February 22-25, 2009, Revised Selected Papers*, volume 5665 of *Lecture Notes in Computer Science*, pages 209–227. Springer, 2009. ISBN 978-3-642-03316-2.

[75] I. ILOG. IBM ILOG CPLEX Optimization Studio V12.7.0 documentation, 2016. Official webpage, https://www-01.ibm.com/software/websphere/products/optimization/cplex-studio-community-edition/.

[76] ISO/IEC 9797-1. Information technology-security techniques-data integrity mechanism using a cryptographic check function employing a block cipher algorithm. international organizatoin for standards.

[77] J. Jean and I. Nikolic. Efficient Design Strategies Based on the AES Round Function. In T. Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 334–353. Springer, 2016. ISBN 978-3-662-52992-8.

[78] J. Jean, I. Nikolic, and T. Peyrin. Tweaks and Keys for Block Ciphers: The TWEAKEY Framework. In P. Sarkar and T. Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings,*

Part II, volume 8874 of *Lecture Notes in Computer Science*, pages 274–288. Springer, 2014. ISBN 978-3-662-45607-1.

[79] J. Jean, I. Nikolic, and T. Peyrin. Tweaks and Keys for Block Ciphers: the TWEAKEY Framework. Cryptology ePrint Archive, Report 2014/831, 2014. http://eprint.iacr.org/2014/831.

[80] Jean, Jérémy and Nikolić, Ivica and Peyrin, Thomas. KIASU v1. Submission to the CAESAR competition:, 2014. http://competitions.cr.yp.to/round1/kiasuv1.pdf.

[81] Jung Hee Cheon, MunJu Kim, and Kwangjo Kim. Impossible Differential Cryptanalysis of Hierocrypt-3 Reduced to 3 Rounds. Nessie report, 2002.

[82] O. Kara. Reflection Cryptanalysis of Some Ciphers. In D. R. Chowdhury, V. Rijmen, and A. Das, editors, *Progress in Cryptology - INDOCRYPT 2008, 9th International Conference on Cryptology in India, Kharagpur, India, December 14-17, 2008. Proceedings*, volume 5365 of *Lecture Notes in Computer Science*, pages 294–307. Springer, 2008. ISBN 978-3-540-89753-8.

[83] O. Kara and C. Manap. A New Class of Weak Keys for Blowfish. In A. Biryukov, editor, *Fast Software Encryption, 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers*, volume 4593 of *Lecture Notes in Computer Science*, pages 167–180. Springer, 2007. ISBN 978-3-540-74617-1.

[84] O. Kazymyrov and V. Kazymyrova. Algebraic Aspects of the Russian Hash Standard GOST R 34.11-2012. Cryptology ePrint Archive, Report 2013/556, 2013. http://eprint.iacr.org/2013/556.

[85] Keccak team. Strengths of Keccak - Design and security. http://keccak.noekeon.org/, Last Accessed: 2014-02-20.

[86] J. Kelsey, T. Kohno, and B. Schneier. Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent. In B. Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, volume 1978 of *Lecture Notes in Computer Science*, pages 75–93. Springer, 2000. ISBN 3-540-41728-1.

[87] L. Knudsen. DEAL: A 128-bit block cipher. *Complexity*, 258(2), 1998. NIST AES Proposal.

[88] L. R. Knudsen. Truncated and Higher Order Differentials. In B. Preneel, editor, *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*, volume 1008 of *Lecture Notes in Computer Science*, pages 196–211. Springer, 1994.

[89] L. R. Knudsen. Block Ciphers. In H. C. A. van Tilborg and S. Jajodia, editors, *Encyclopedia of Cryptography and Security, 2nd Ed.*, pages 152–157. Springer, 2011. ISBN 978-1-4419-5905-8.

[90] L. R. Knudsen and J. E. Mathiassen. A Chosen-Plaintext Linear Attack on DES. In B. Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, volume 1978 of *Lecture Notes in Computer Science*, pages 262–272. Springer, 2000. ISBN 3-540-41728-1.

[91] L. R. Knudsen and V. Rijmen. Known-Key Distinguishers for Some Block Ciphers. In K. Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, volume 4833 of *Lecture Notes in Computer Science*, pages 315–324. Springer, 2007. ISBN 978-3-540-76899-9.

[92] L. R. Knudsen and M. Robshaw. *The Block Cipher Companion*. Information Security and Cryptography. Springer, 2011. ISBN 978-3-642-17341-7.

[93] L. R. Knudsen and D. A. Wagner. Integral Cryptanalysis. In J. Daemen and V. Rijmen, editors, *Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers*, volume 2365 of *Lecture Notes in Computer Science*, pages 112–127. Springer, 2002. ISBN 3-540-44009-7.

[94] X. Lai. Higher Order Derivatives and Differential Cryptanalysis. In R. Blahut, J. Costello, DanielJ., U. Maurer, and T. Mittelholzer, editors, *Communications and Cryptography*, volume 276 of *The Springer International Series in Engineering and Computer Science*, pages 227–233. Springer US, 1994. ISBN 978-1-4613-6159-6.

[95] X. Lai and J. L. Massey. A Proposal for a New Block Encryption Standard. In I. Damgård, editor, *Advances in Cryptology - EUROCRYPT '90, Workshop on the Theory and Application of of Cryptographic Techniques, Aarhus, Denmark, May 21-24, 1990, Proceedings*, volume 473 of *Lecture Notes in Computer Science*, pages 389–404. Springer, 1990. ISBN 3-540-53587-X.

[96] X. Lai, J. L. Massey, and S. Murphy. Markov Ciphers and Differential Cryptanalysis. In D. W. Davies, editor, *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, volume 547 of *Lecture Notes in Computer Science*, pages 17–38. Springer, 1991. ISBN 3-540-54620-0.

[97] S. K. Langford and M. E. Hellman. Differential-Linear Cryptanalysis. In Y. Desmedt, editor, *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, volume 839 of *Lecture Notes in Computer Science*, pages 17–25. Springer, 1994. ISBN 3-540-58333-5.

[98] G. Leurent. Practical Key Recovery Attack against Secret-IV Edon-R. In J. Pieprzyk, editor, *Topics in Cryptology - CT-RSA 2010, The Cryptographers' Track at the RSA Con-*

ference 2010, San Francisco, CA, USA, March 1-5, 2010. Proceedings, volume 5985 of *Lecture Notes in Computer Science*, pages 334–349. Springer, 2010. ISBN 978-3-642-11924-8.

[99] L. Li, K. Jia, and X. Wang. Improved Meet-in-the-Middle Attacks on AES-192 and PRINCE. Cryptology ePrint Archive, Report 2013/573, 2013. http://eprint.iacr.org/2013/573.

[100] R. Li and C. Jin. Meet-in-the-middle attacks on 10-round AES-256. *Des. Codes Cryptography*, 80(3):459–471, 2016.

[101] M. Liskov, R. L. Rivest, and D. A. Wagner. Tweakable Block Ciphers. *J. Cryptology*, 24 (3):588–613, 2011.

[102] Logic Friday. http://sontrak.com/.

[103] J. Lu, O. Dunkelman, N. Keller, and J. Kim. New Impossible Differential Attacks on AES. In D. R. Chowdhury, V. Rijmen, and A. Das, editors, *Progress in Cryptology - INDOCRYPT 2008, 9th International Conference on Cryptology in India, Kharagpur, India, December 14-17, 2008. Proceedings*, volume 5365 of *Lecture Notes in Computer Science*, pages 279–293. Springer, 2008. ISBN 978-3-540-89753-8.

[104] J. Lu, J. Kim, N. Keller, and O. Dunkelman. Improving the Efficiency of Impossible Differential Cryptanalysis of Reduced Camellia and MISTY1. In T. Malkin, editor, *Topics in Cryptology - CT-RSA 2008, The Cryptographers' Track at the RSA Conference 2008, San Francisco, CA, USA, April 8-11, 2008. Proceedings*, volume 4964 of *Lecture Notes in Computer Science*, pages 370–386. Springer, 2008. ISBN 978-3-540-79262-8.

[105] B. Ma, B. Li, R. Hao, and X. Li. Improved Cryptanalysis on Reduced-Round GOST and Whirlpool Hash Function. In I. Boureanu, P. Owesarski, and S. Vaudenay, editors, *Applied Cryptography and Network Security - 12th International Conference, ACNS 2014,*

Lausanne, Switzerland, June 10-13, 2014. Proceedings, volume 8479 of *Lecture Notes in Computer Science*, pages 289–307. Springer, 2014. ISBN 978-3-319-07535-8.

[106] F. J. MacWilliams and N. J. A. Sloane. *The theory of error correcting codes*, volume 16. Elsevier, 1977.

[107] H. Mala, M. Dakhilalian, V. Rijmen, and M. Modarres-Hashemi. Improved Impossible Differential Cryptanalysis of 7-Round AES-128. In G. Gong and K. C. Gupta, editors, *Progress in Cryptology - INDOCRYPT 2010 - 11th International Conference on Cryptology in India, Hyderabad, India, December 12-15, 2010. Proceedings*, volume 6498 of *Lecture Notes in Computer Science*, pages 282–291. Springer, 2010. ISBN 978-3-642-17400-1.

[108] M. Matsui. Linear Cryptanalysis Method for DES Cipher. In T. Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer, 1993. ISBN 3-540-57600-2.

[109] D. Matyukhin and V. Shishkin. Some methods of hash functions analysis with application to the GOST P 34.11-94 algorithm. *Mat. Vopr. Kriptogr*, 3:71–89, 2012.

[110] F. Mendel, N. Pramstaller, and C. Rechberger. A (Second) Preimage Attack on the GOST Hash Function. In K. Nyberg, editor, *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, volume 5086 of *Lecture Notes in Computer Science*, pages 224–234. Springer, 2008. ISBN 978-3-540-71038-7.

[111] F. Mendel, C. Rechberger, M. Schläffer, and S. S. Thomsen. The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In O. Dunkelman, editor, *Fast Software Encryption, 16th International Workshop, FSE 2009, Leuven, Belgium, February*

22-25, 2009, Revised Selected Papers, volume 5665 of *Lecture Notes in Computer Science*, pages 260–276. Springer, 2009. ISBN 978-3-642-03316-2.

[112] A. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. ISBN 0-8493-8523-7.

[113] N. Mouha, Q. Wang, D. Gu, and B. Preneel. Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In C. Wu, M. Yung, and D. Lin, editors, *Information Security and Cryptology - 7th International Conference, Inscrypt 2011, Beijing, China, November 30 - December 3, 2011. Revised Selected Papers*, volume 7537 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2011. ISBN 978-3-642-34703-0.

[114] Y. D. Mulder. *White-Box Cryptography: Analysis of White-Box AES Implementations (White-Box Cryptografie: Analyse van White-Box AES implementaties)*. PhD thesis, Katholieke Universiteit Leuven, Belgium, 2014.

[115] National Institute of Standards and Technology. *Federal Information Processing Standards Publication 197: Advanced Encryption Standard (AES)*. NIST, November 2001.

[116] New European Schemes for Signatures, Integrity, and Encryption. https://www.cosic.esat.kuleuven.be/nessie.

[117] NIST Special Publication 800-57 Part 1 Revision 4. Recommendation for Key Management Part 1: General, 2016. http://dx.doi.org/10.6028/NIST.SP.800-57pt1r4.

[118] K. Nyberg. Linear Approximation of Block Ciphers. In A. D. Santis, editor, *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 439–444. Springer, 1994. ISBN 3-540-60176-7.

[119] P. Oechslin. Making a Faster Cryptanalytic Time-Memory Trade-Off. In D. Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 617–630. Springer, 2003. ISBN 3-540-40674-3.

[120] K. Ohkuma, H. Muratani, F. Sano, and S. Kawamura. The Block Cipher Hierocrypt. In D. R. Stinson and S. E. Tavares, editors, *Selected Areas in Cryptography, 7th Annual International Workshop, SAC 2000, Waterloo, Ontario, Canada, August 14-15, 2000, Proceedings*, volume 2012 of *Lecture Notes in Computer Science*, pages 72–88. Springer, 2000. ISBN 3-540-42069-X.

[121] K. Ohkuma, F. Sano, H. Muratani, M. Motoyama, and S. Kawamura. On Security of Block Ciphers Hierocrypt-3 and Hierocrypt-L1. The 2001 Symposium on Cryptography and Information Security (SCIS 2001), 11A-4, Jan. 2001.

[122] B. Preneel. Hash Functions. In H. C. A. van Tilborg and S. Jajodia, editors, *Encyclopedia of Cryptography and Security, 2nd Ed.*, pages 543–553. Springer, 2011. ISBN 978-1-4419-5905-8.

[123] B. Preneel. MAC Algorithms. In H. C. A. van Tilborg and S. Jajodia, editors, *Encyclopedia of Cryptography and Security, 2nd Ed.*, pages 742–748. Springer, 2011. ISBN 978-1-4419-5905-8.

[124] B. Preneel and P. C. van Oorschot. On the Security of Iterated Message Authentication Codes. *IEEE Trans. Information Theory*, 45(1):188–199, 1999.

[125] C. Rechberger. Security evaluation of 128-bit block ciphers AES, CIPHERUNICORN-A, and Hierocrypt-3 against biclique attacks, 2012.

[126] V. Rijmen. *Cryptanalysis and Design of Iterated Block Ciphers*. PhD thesis, Doctoral Dissertation, October 1997, KU Leuven, 1997.

[127] Y. Sasaki and Y. Todo. New Impossible Differential Search Tool from Design and Cryptanalysis Aspects - Revealing Structural Properties of Several Ciphers. In J. Coron and J. B. Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III*, volume 10212 of *Lecture Notes in Computer Science*, pages 185–215, 2017. ISBN 978-3-319-56616-0.

[128] S.Furuya and V.Rijmen. Observations on Hierocrypt-3/L1 key-scheduling algorithms. 2nd NESSIE Workshop, 2001.

[129] C. E. Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 28(4):656–715, 1949.

[130] L. Sun, W. Wang, R. Liu, and M. Wang. MILP-Aided Bit-Based Division Property for ARX-Based Block Cipher. Cryptology ePrint Archive, Report 2016/1101, 2016. http://eprint.iacr.org/2016/1101.

[131] L. Sun, W. Wang, and M. Wang. MILP-Aided Bit-Based Division Property for Primitives with Non-Bit-Permutation Linear Layers. Cryptology ePrint Archive, Report 2016/811, 2016. http://eprint.iacr.org/2016/811.

[132] S. Sun, D. Gerault, P. Lafourcade, Q. Yang, Y. Todo, K. Qiao, and L. Hu. Analysis of AES, SKINNY, and Others with Constraint Programming. *IACR Trans. Symmetric Cryptol.*, 2017(1):281–306, 2017.

[133] S. Sun, L. Hu, M. Wang, P. Wang, K. Qiao, X. Ma, D. Shi, L. Song, and K. Fu. Towards Finding the Best Characteristics of Some Bit-oriented Block Ciphers and Automatic Enumeration of (Related-key) Differential and Linear Characteristics with Predefined Properties. Cryptology ePrint Archive, Report 2014/747, 2014. http://eprint.iacr.org/2014/747.

[134] S. Sun, L. Hu, P. Wang, K. Qiao, X. Ma, and L. Song. Automatic Security Evaluation and (Related-key) Differential Characteristic Search: Application to SIMON, PRESENT, LBlock, DES(L) and Other Bit-Oriented Block Ciphers. In P. Sarkar and T. Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 158–178. Springer, 2014. ISBN 978-3-662-45610-1.

[135] T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi. TWINE: A Lightweight Block Cipher for Multiple Platforms. In L. R. Knudsen and H. Wu, editors, *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, volume 7707 of *Lecture Notes in Computer Science*, pages 339–354. Springer, 2012. ISBN 978-3-642-35998-9.

[136] B. Taga, S. Moriai, and K. Aoki. Differential and Impossible Differential Related-Key Attacks on Hierocrypt-L1. In W. Susilo and Y. Mu, editors, *Information Security and Privacy - 19th Australasian Conference, ACISP 2014, Wollongong, NSW, Australia, July 7-9, 2014. Proceedings*, volume 8544 of *Lecture Notes in Computer Science*, pages 17–33. Springer, 2014. ISBN 978-3-319-08343-8.

[137] C. Tezcan. Improbable differential attacks on Present using undisturbed bits. *J. Computational Applied Mathematics*, 259:503–511, 2014.

[138] The National Hash Standard of the Russian Federation GOST R 34.11-2012. Russian Federal Agency on Technical Regulation and Metrology report, 2012. https://www.tc26.ru/en/GOSTR3411\discretionary{-}{}{}2012/GOST_R_34_11\discretionary{-}{}{}2012_eng.pdf.

[139] H. C. A. V. Tilborg. *Fundamentals of Cryptology: A Professional Reference and Interactive Tutorial*. Kluwer Academic Publishers, 1st edition, 1999. ISBN 0792386752.

[140] Y. Todo. Structural Evaluation by Generalized Integral Property. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 287–314. Springer, 2015. ISBN 978-3-662-46799-2.

[141] M. Tolba, A. Abdelkhalek, and A. M. Youssef. Meet-in-the-Middle Attacks on Reduced Round Piccolo. In T. Güneysu, G. Leander, and A. Moradi, editors, *Lightweight Cryptography for Security and Privacy - 4th International Workshop, LightSec 2015, Bochum, Germany, September 10-11, 2015, Revised Selected Papers*, volume 9542 of *Lecture Notes in Computer Science*, pages 3–20. Springer, 2015. ISBN 978-3-319-29077-5.

[142] M. Tolba, A. Abdelkhalek, and A. M. Youssef. Meet-in-the-Middle Attacks on Round-Reduced Khudra. In R. S. Chakraborty, P. Schwabe, and J. A. Solworth, editors, *Security, Privacy, and Applied Cryptography Engineering - 5th International Conference, SPACE 2015, Jaipur, India, October 3-7, 2015, Proceedings*, volume 9354 of *Lecture Notes in Computer Science*, pages 127–138. Springer, 2015. ISBN 978-3-319-24125-8.

[143] M. Tolba, A. Abdelkhalek, and A. M. Youssef. A Meet in the Middle Attack on Reduced Round Kiasu-BC. *IEICE Transactions*, 99-A(10):1888–1890, 2016.

[144] M. Tolba, A. Abdelkhalek, and A. M. Youssef. Truncated and Multiple Differential Cryptanalysis of Reduced Round Midori128. In M. Bishop and A. C. A. Nascimento, editors, *Information Security - 19th International Conference, ISC 2016, Honolulu, HI, USA, September 3-6, 2016, Proceedings*, volume 9866 of *Lecture Notes in Computer Science*, pages 3–17. Springer, 2016. ISBN 978-3-319-45870-0.

[145] M. Tolba, A. Abdelkhalek, and A. M. Youssef. Impossible Differential Cryptanalysis of Reduced-Round SKINNY. In M. Joye and A. Nitaj, editors, *Progress in Cryptology - AFRICACRYPT 2017 - 9th International Conference on Cryptology in Africa, Dakar,*

Senegal, May 24-26, 2017, Proceedings, volume 10239 of *Lecture Notes in Computer Science*, pages 117–134, 2017. ISBN 978-3-319-57338-0.

[146] M. Tolba, A. Abdelkhalek, and A. M. Youssef. Improved Multiple Impossible Differential Cryptanalysis of Midori128. *IEICE Transactions*, 100-A(8):1733–1737, 2017.

[147] M. Tolba, A. Abdelkhalek, and A. M. Youssef. Multidimensional Zero-Correlation Linear Cryptanalysis of Reduced Round SPARX-128. In J. Camenisch and C. Adams, editors, *Selected Areas in Cryptography - SAC 2017 - 24th International Conference, Ottawa, ON, Canada, August 16-18, 2017, Revised Selected Papers*, Lecture Notes in Computer Science. Springer, 2017. to appear.

[148] Toshiba Corporation. Block Cipher Family Hierocrypt. http://www.toshiba.co.jp/rdc/security/hierocrypt/index.htm.

[149] S. Vaudenay. Decorrelation: A Theory for Block Cipher Security. *J. Cryptology*, 16(4): 249–286, 2003.

[150] D. A. Wagner. Towards a Unifying View of Block Cipher Cryptanalysis. In B. K. Roy and W. Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 16–33. Springer, 2004. ISBN 3-540-22171-9.

[151] Z. Wang, H. Yu, and X. Wang. Cryptanalysis of GOST R hash function. *Inf. Process. Lett.*, 114(12):655–662, 2014.

[152] S. Wu and M. Wang. Security Evaluation against Differential Cryptanalysis for Block Cipher Structures. Cryptology ePrint Archive, Report 2011/551, 2011. http://eprint.iacr.org/2011/551.

[153] Z. Xiang, W. Zhang, Z. Bao, and D. Lin. Applying MILP Method to Searching Integral Distinguishers Based on Division Property for 6 Lightweight Block Ciphers. In J. H.

Cheon and T. Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 648–678, 2016. ISBN 978-3-662-53886-9.

[154] W. Zhang and V. Rijmen. Division Cryptanalysis of Block Ciphers with a Binary Diffusion Layer. Cryptology ePrint Archive, Report 2017/188, 2017. http://eprint.iacr.org/2017/188.

[155] J. Zou, W. Wu, and S. Wu. Cryptanalysis of the Round-Reduced GOST Hash Function. In D. Lin, S. Xu, and M. Yung, editors, *Information Security and Cryptology - 9th International Conference, Inscrypt 2013, Guangzhou, China, November 27-30, 2013, Revised Selected Papers*, volume 8567 of *Lecture Notes in Computer Science*, pages 309–322. Springer, 2013. ISBN 978-3-319-12086-7.