Cryptanalysis of Block Ciphers with New Design Strategies

Mohamed Tolba

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, Quebec, Canada

October 2017

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Mohamed Tolba

Entitled: Cryptanalysis of Block Ciphers with New Design Strategies

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. Theodore Stathopoulos

External Examiner
Dr. Huapeng Wu

External to Program
Dr. Anjali Agarwal

Examiner
Dr. Lingyu Wang

Examiner
Dr. Mohammad Mannan

Thesis Supervisor
Dr. Amr M. Youssef

Approved by

Dr. Chadi Assi, Graduate Program Director

December 4^{th},2017

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

Abstract

Cryptanalysis of Block Ciphers with New Design Strategies

Mohamed Tolba, Ph.D.

Concordia University, 2017

Block ciphers are among the mostly widely used symmetric-key cryptographic primitives, which are fundamental building blocks in cryptographic/security systems. Most of the public-key primitives are based on hard mathematical problems such as the integer factorization in the RSA algorithm and discrete logarithm problem in the DiffieHellman. Therefore, their security are mathematically proven. In contrast, symmetric-key primitives are usually not constructed based on well-defined hard mathematical problems. Hence, in order to get some assurance in their claimed security properties, they must be studied against different types of cryptanalytic techniques. Our research is dedicated to the cryptanalysis of block ciphers. In particular, throughout this thesis, we investigate the security of some block ciphers constructed with new design strategies. These new strategies include (i) employing simple round function, and modest key schedule, (ii) using another input called *tweak* rather than the usual two inputs of the block ciphers, the plaintext and the key, to instantiate different permutations for the same key. This type of block ciphers is called a tweakable block cipher, (iii) employing linear and non-linear components that are energy efficient to provide low energy consumption block ciphers, (iv) employing optimal diffusion linear transformation layer while following the AES-based construction to provide faster diffusion rate, and (v) using rather weak but larger S-boxes in addition to simple linear transformation layers to provide provable security of ARX-based block ciphers against single characteristic differential and linear cryptanalysis. The results presented in this thesis can be summarized as follows:

Initially, we analyze the security of two lightweight block ciphers, namely, Khudra and Piccolo against Meet-in-the-Middle (MitM) attack based on the Demirci and Selçuk approach exploiting the simple design of the key schedule and round function.

Next, we investigate the security of two tweakable block ciphers, namely, Kiasu-BC and SKINNY. According to the designers, the best attack on Kiasu-BC covers 7 rounds. However, we exploited the tweak to present 8-round attack using MitM with efficient enumeration cryptanalysis. Then, we improve the previous results of the impossible differential cryptanalysis on SKINNY exploiting the tweakey schedule and linear transformation layer.

Afterwards, we study the security of new low energy consumption block cipher, namely, Midori128 where we present the longest impossible differential distinguishers that cover complete 7 rounds. Then, we utilized 4 of these distinguishers to launch key recovery attack against 11 rounds of Midori128 to improve the previous results on this cipher using the impossible differential cryptanalysis. Then, using the truncated differential cryptanalysis, we are able to attack 13 rounds of Midori128 utilizing a 10-round differential distinguisher.

We also analyze Kuznyechik, the standard Russian federation block cipher, against MitM with efficient enumeration cryptanalysis where we improve the previous results on Kuznyechik, using MitM attack with efficient enumeration, by presenting 6-round attack. Unlike the previous attack, our attack exploits the exact values of the coefficients of the MDS transformation that is used in the cipher.

Finally, we present key recovery attacks using the multidimensional zero-correlation cryptanalysis against SPARX-128, which follows the long trail design strategy, to provide provable security of ARX-based block ciphers against single characteristic differential and linear cryptanalysis.

Acknowledgments

First and foremost, I would like to express my special appreciation and sincere gratitude to my supervisor, Dr. Amr Youssef, for his continuous support, motivation, patience, enthusiasm, and knowledge that helped me to finish this work. His willingness to give his time so generously has been very much appreciated. I appreciate your invaluable advices you give me on both research and my career.

Next, I would like to thank my colleagues in the CIISE Crypto Lab for their friendship and support. Special thanks to Ahmed Abdelkhalek for the long hours we spent together in discussing our research problems.

Finally, many grateful thanks for my lovely wife for her love, support, and encouragement during my PhD study. Special thanks to my mother, my mother and father in-laws for their support, encouragement, and love.

Mohamed Tolba

To my family for their love and support

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 General Overview and Motivation

Cryptography is the study of mathematical techniques to achieve information security. The aim of cryptography [105] is to help ensure the following security goals: (i) confidentiality, which keeps the content of information from all but those authorized to have it, (ii) data integrity, which addresses the unauthorized alteration of data, (iii) authentication, which addresses the identification, and it has two major classes: entity authentication and data origin authentication, and (iv) non-repudiation, which prevents an entity from denying previous commitments or actions.

The above goals can be obtained through the use of various primitives such as block ciphers, hash functions, Message Authentication Codes (MACs), and digital signatures. As illustrated in Figure 1.1, cryptographic primitives can be classified into three categories, namely, unkeyed primitives, symmetric-key primitives, and public-key primitives. For unkeyed primitives, no secret information is used. Hash functions are important examples of these primitives where their main functionality is to provide data integrity. Nowadays, there are a lot of efficient and secure hash functions such as Keccak, the winner of the SHA-3 competition [25], the Russian hash function standard Streebog [4], and the Ukraine hash function standard Kupyna [111], to name a few.

Symmetric-key primitives are another type of cryptographic primitives, where a single secret key is shared between the communicating entities. These types of primitives include the following primitives: block ciphers where the secret key is used to map a data block of fixed length, called plaintext, to a data block of the same length, called ciphertext, through an encryption algorithm; and map the ciphertext to plaintext through a decryption algorithm. The Advanced Encryption Standard (AES) is an example of symmetric-key block ciphers.

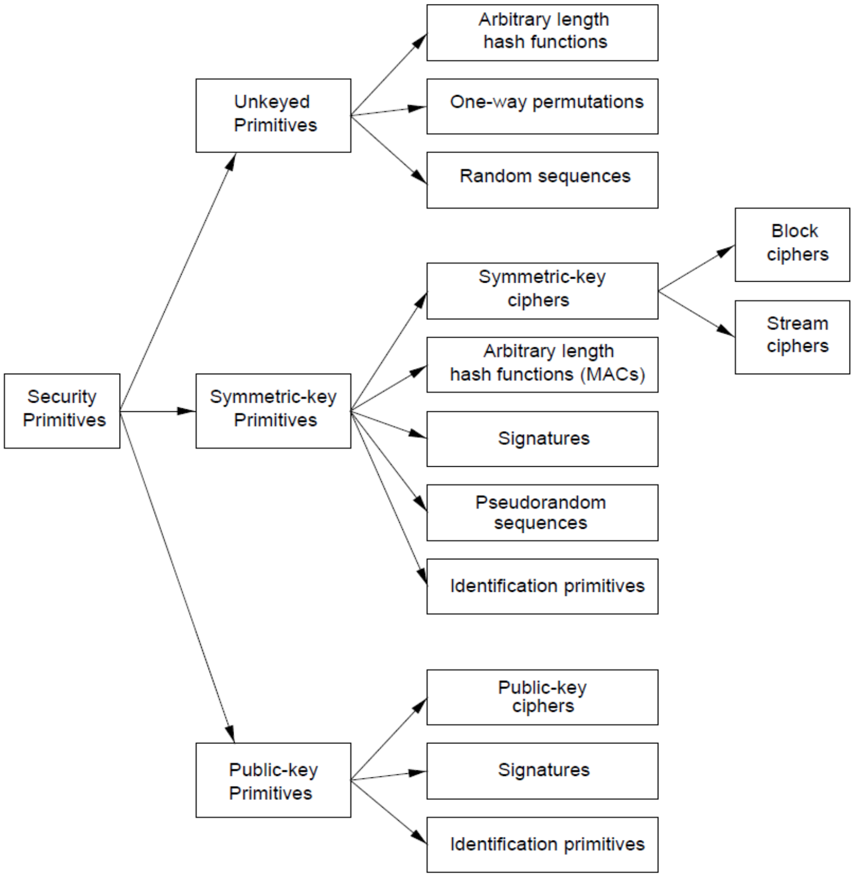


Figure 1.1: A taxonomy of cryptographic primitives [105]

MACs, which provide data origin authentication, are another type of symmetric-key primitives and can be constructed using keyed hash functions. MAC schemes can also be based on block ciphers instead of hash functions, but hash functions-based MACs are usually faster than block ciphers-based ones.

Public-key primitives differ from symmetric-key primitives in that every communicating party has two different keys, namely public and private keys, where the public key is globally known to everyone and the private key is kept secret. Given the public key, it is infeasible to compute the corresponding private key, however the public key is easily computable from its corresponding private key. The public key should be related to the entity that has its corresponding private key through a certificate issued by a trusted authority. The RSA algorithm is an example of public-key encryption schemes.

When using symmetric-key primitives, there is a need for key management schemes, especially in larger networks because every two communicating parties should have their private key, and every time they establish connection they need another secret key. On the other hand, compared to public-key primitives, symmetric-key primitives have much higher throughput. Therefore, in practice, to acquire the advantages of the two primitives, public-key primitives are used to establish session keys that are utilized for encryption and decryption using symmetric-key primitives.

Cryptanalysis is the scientific discipline of dissecting and studying the security of cryptographic primitives aiming to discover weaknesses that can result in violating the security aspects provided by the primitives. Most of the public-key primitives are based on hard mathematical problems such as the integer factorization in the RSA algorithm and the discrete logarithm problem in DiffieHellman schemes. Therefore, their security are mathematically proven. In contrast, symmetric-key primitives do not depend on such hard mathematical problems. Hence, to get some assurance in their claimed security properties, they must be carefully analyzed against different types of cryptanalytic techniques that will be mentioned in the next chapter.

Symmetric-key block ciphers can be used to build other primitives such as hash functions, MAC schemes, stream ciphers, and Authenticated Encryption (AE) schemes. Moreover, block ciphers are basic component in any cryptosystem (a cryptosystem is system incorporating a number of primitives to provide a more complex solution), and as mentioned above, their security are not mathematically proven as in the public-key primitives. In addition, due to the rapid increase of the development of resource constrained devices such as RFIDs and wireless sensor networks, several lightweight block ciphers were recently proposed in order to be deployed on these resource constrained devices. These lightweight block ciphers usually have shorter key length, simple key schedules, and more compact round functions.

Therefore, the security of these block ciphers, whether they are conventional or lightweight, should be well studied because any weakness can result in breaking the cryptosystems that are using them. Hence, throughout this thesis, we focus on the analysis of some block ciphers that employ new design strategies. Some of these block ciphers were presented at the NIST lightweight cryptography workshop in addition to top tier conferences such as CRYPTO, ASIACRYPT and CHES. Furthermore, these block ciphers are expected to be deployed in resource constrained devices (including IoT devices) in the near future. On the other hand, these new strategies include (i) the use of more compact round function and key schedule in order to be deployed on resource constrained devices (lightweight block ciphers), (ii) the use of another input called tweak, to instantiate different permutations for the same key (tweakable block ciphers), (iii) the use of low energy linear and non-linear layers (low energy consumption block ciphers), (iv) the use of optimal diffusion linear transformation layers (high diffusion rate block ciphers), and (v) the use of large S-boxes in addition to simple linear layer, to provide provable security against differential and linear cryptanalysis for ARX-based block ciphers.

1.2 Thesis Contributions

In this thesis, we analyze some block ciphers that employ new design strategies against a list of advances cryptanalytic attacks. These ciphers include the lightweight block ciphers Khudra and Piccolo; the tweakable block ciphers Kiasu-BC and SKINNY; the low energy block cipher Midori128; the high diffusion rate block cipher Kuznyechik; and the long trail strategy-based cipher SPARX-128. Our contributions can be summarized as follows:

- We study the implication of using more compact round function in addition to simple key schedule by analyzing two lightweight block ciphers, namely, Khudra and Piccolo using MitM attack based on Demirci and Selçuk approach (Plain MitM).

- We investigate the security of the tweakable block cipher Kiasu-BC, where we exploit the additional input tweak to present 8-round attack using MitM with efficient enumeration technique.

- We mount key recovery attacks against all the 6 variants of SKINNY family of lightweight tweakable block ciphers. More precisely, we exploit the properties of the mix column operation and the simple tweakey schedule to launch impossible differential attacks against all the variants of SKINNY.

- We analyze the strength of the low energy consumption block cipher Midori128 against two different attacks. In particular, we propose the longest impossible differential distinguisher that covers complete 7 rounds, including the mix column operations, of Midori128. Then, we exploit the existence of multiple such distinguishers to launch a key recovery attack against 11 rounds of Midori128 using multiple impossible differential cryptanalysis. Then, using the differential cryptanalysis, we propose 10-round differential distinguishers that are exploited to mount a 13-round key recovery attack against Midori128 using truncated differential cryptanalysis.

- We study the security margin of the standard Russian block cipher Kuznyechik. In particular, we show that there exists a 5-round attack in the chosen plaintext model. Then, using the exact values of the coefficient of the mix column operation, we are able to present a 6-round attack using MitM with efficient enumeration cryptanalysis technique.

- We analyze the long trail design strategy employed in the SPARX family of ARX-based block ciphers. More precisely, we propose 20/21-round distinguishers against

SPARX-128 utilizing the zero-correlation property. Then, we exploit these distinguishers to launch key recovery attacks against SPARX-128 using the multidimensional zero correlation cryptanalysis technique.

The above contributions have been published in [126, 127, 128, 129, 130, 131, 132, 133]. Other works conducted during the tenure of this Ph.D. have been published in [6, 7, 8, 9, 134].

Chapter 2

Background

In this chapter, we present a brief description of block ciphers and their constructions. Then, we discuss different attack models that are utilized in the analysis of block ciphers. Finally, we review some of the cryptanalytic techniques that can be utilized to mount attacks on block ciphers such as differential, linear, differential-linear, higher-order differential, integral, truncated, impossible, and Meet-in-the-Middle cryptanalytic techniques.

2.1 Block Ciphers

A block cipher E is a keyed permutation. More specifically, it is a bijective mapping from the plaintext $\mathbf{P}$ to the ciphertext $\mathbf{C}$. Applying this mapping E is called *encryption* and applying the inverse of E is called *decryption*. More formally, a block cipher is defined as follows:

Definition 1 ***Block cipher*** [37]. A mapping $E : \mathbb{F}_2^n \times \mathbb{F}_2^k \rightarrow \mathbb{F}_2^n$ is called a block cipher with block size n bits and key size k bits, if the mapping $E(., K)$ is a bijection for each $K \in \mathbb{F}_2^k$, that is, if the inverse mapping $E^{-1}(., K)$ exists with $E^{-1}(E(K, x), K) = x$ for each $K \in \mathbb{F}_2^k$ and $x \in \mathbb{F}_2^n$.

A block cipher has 2^k permutations, i.e., each key corresponds to one permutation, while for a block length of n bits, we have $2^n!$ permutations. Therefore, for a block cipher to be ideal, its permutations should be chosen randomly from $2^n!$. More formally, an ideal block cipher is defined as follows:

Definition 2 ***Ideal block cipher*** [37]. A block cipher E is called ideal, if E is defined by assigning a random element of the symmetric group $\mathbf{S}_{\mathbb{F}_2^n}$ to each of the 2^k permutations $E(., K)$.

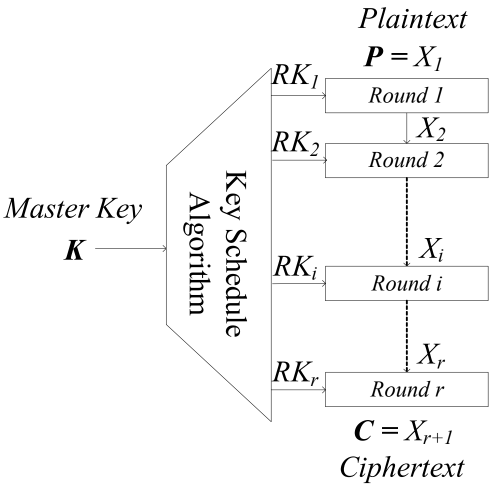
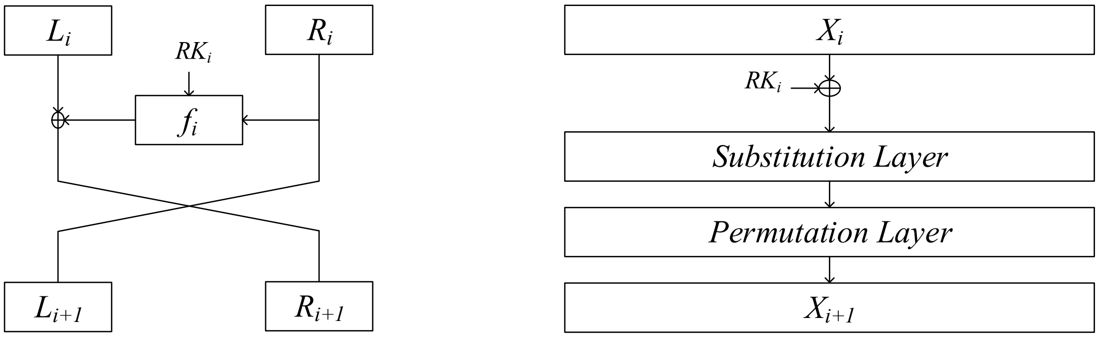


Figure 2.1: Iterative block ciphers

2.1.1 Block Cipher Design

Currently, most of the newly proposed block ciphers have a block length of 128 bits and a key length of 128/192/256 bits, while newly proposed lightweight block ciphers have a block length of 64 bits and key lengths of 64/80/92/128 bits. As illustrated in Figure 2.1, most of the block ciphers are implemented by applying a round function (f) r times, this what we call iterative block ciphers, where in round i, the round function f_i updates the internal state X_i using the round key RK_i to obtain a new state X_{i+1}. The round keys RK_is are generated from the master key K using the key schedule algorithm.

There are two main designs of block ciphers, namely Feistel Network (FN) and Substitution Permutation Network (SPN). In the FN ciphers, the data block is split into two halves, namely L_i, R_i, as illustrated in Figure 2.2a. Then, the round function is applied to the right half R_i. Then, the output of the round function $f_i(R_i, RK_i)$ is XORed with the left half L_i. Finally, the resulting two halves are swapped. DES [2] is an example of the FN ciphers. While, in the SPN ciphers the round function is applied to the whole data block. First, the data block X_i is XORed with the round key RK_i; this operation is called the key addition. Second, a non-linear layer called the substitution layer is applied to the output of the key addition layer $X_i \oplus RK_i$. Finally, a linear permutation is applied to the output of the previous layer as shown in Figure 2.2b. AES [3] is an example of SPN block ciphers. The major difference between the FN and SPN ciphers is that the round function in the FN can be non bijective while the round function in the SPN ciphers should be bijective in order to have unique decryption.



(a) Feistel Network (b) Substitution Permutation Network

Figure 2.2: Approaches for constructing block ciphers

The round function in the iterative block ciphers consists of two categories of layers, namely linear and non-linear layers. The linear layer contains XOR addition, bit rotations, or permutations. While the non-linear layer contains operations such as substitution boxes, addition modulo 2^n, or multiplication modulo 2^n. By employing these simple operations in the round function and iterating the round function r rounds, we achieve the required mixing between the plaintext and key in order to achieve high security. The branch number can be used to evaluate the diffusion power of the linear transformation.

Definition 3 [99] Suppose that $B = \{0,1\}^8$ and $\mathbb{N} = \{0,1,2,3,\cdots\}$. Let $W : B^* \rightarrow \mathbb{N}$ be the function returning the number of non-zero bytes of an input byte tuple. The branch number of a linear transformation $L : B^m \rightarrow B^n$ (for specific values of m and n) is defined to equal the minimum value of $W(x) + W(L(x))$, where $x \in B^m - \{0^m\}$.

2.1.2 Block Cipher Evaluation

According to [105], block ciphers can be assessed in practice using the following criteria:

1. Estimated security level. Confidence in the (historical) security of a cipher grows if it has been subjected to and withstood expert cryptanalysis over a substantial time period; such ciphers are certainly considered more secure than those which have not.

2. Key size. The effective bit length of the key, or more specifically, the entropy of the key space, defines an upper bound on the security of a cipher (by considering exhaustive search). Longer keys typically impose additional costs (e.g., generation, transmission, storage, and difficulty to remember passwords).

3. Throughput. Throughput is related to the complexity of the cryptographic mapping (see below), and the degree to which the mapping is tailored to a particular implementation medium or platform.

4. Block size. Block size impacts both security (larger is desirable) and complexity (larger is more costly to implement). Block size may also affect performance, for example, if padding is required.

5. Complexity of cryptographic mapping. Algorithmic complexity affects the implementation costs both in terms of development and fixed resources (hardware gate count or software code/data size), as well as real-time performance for fixed resources (throughput). Some ciphers specifically favor hardware or software implementations.

6. Data expansion. It is generally desirable, and often mandatory, that encryption does not increase the size of plaintext data. Randomized encryption techniques result in data expansion.

7. Error propagation. Decryption of ciphertext containing bit errors may result in various effects on the recovered plaintext, including propagation of errors to subsequent plaintext blocks. Different error characteristics are acceptable in various applications. Block size (above) typically affects error propagation.

2.2 Block Cipher Security

Cryptanalysis is the complementary discipline of the cryptography. It is important to evaluate the security of block ciphers to determine their effective security margin. According to [105], attacks on block ciphers can result in a total break (given subset of the plaintext/ciphertext pairs, the key can be recovered by the adversary) or in a partial break (given the ciphertext, the plaintext can be recovered by the adversary) of these ciphers.

The security of the block ciphers can be evaluated assuming the following [105]: the data is transmitted over non-secure channel (thus, the adversary can know all the data transmitted over this channel) and the only secret in the encryption system is the key (i.e., all the details of the encryption system are accessible by the adversary) .

In the context of symmetric-key ciphers, a cryptanalytic attack is an algorithm that tries to retrieve part of or the whole secret key, or even to distinguish between a block cipher and a random permutation. Therefore, we use the following criteria to determine the effectiveness of different attacks [99]: data complexity (the numbers of plaintexts and/or ciphertexts required for execution of the attack), memory complexity (the amount of memory required for execution of the attack), and time complexity (the amount of computation or time required for execution of the attack).

2.2.1 Attack Models

Attack models can also be classified based on the information available to the eavesdropper and assumptions about her capabilities[105]:

1. Ciphertext-only: Only the ciphertext is available.

2. Known-plaintext: Plaintext/ciphertext pairs are available.

3. Chosen-plaintext: The adversary has access to the ciphertexts corresponding to the plaintexts she chooses.

4. Adaptive chosen-plaintext: The adversary may be choose a plaintext depending on a previously chosen-plaintext.

5. Chosen-ciphertext: The adversary has access to the plaintexts of the ciphertexts she chooses.

6. Adaptive chosen-ciphertext: The adversary may choose a ciphertext depending on a previously chosen-ciphertext.

2.2.2 Generic Attacks

Generic attacks are attacks that can be applied on any block cipher of block length n bits and key length k bits regardless of the structure of the cipher. In what follows we describe three of these attacks [99]:

1. Dictionary attack: This attack can deduce the key that is used to encrypt a specific plaintext. This is served by constructing a table containing 2^k ciphertexts for specific plaintext using all the 2^k possible keys. This attack can deduce the unique key when $k < n$, while in the case of $k > n$, it recovers 2^{k-n} key candidates. This attack requires 2^k pre-computations, 2^k ciphertexts, and 2^k n-bit memory storage.

2. Codebook attack: This attack is different from the previous attack in that, all the possible 2^n plaintexts are encrypted using specific (unknown) key and stored in a table indexed by the ciphertext. Then, the attacker can deduce the plaintext of any obtained ciphertext, given that it is encrypted using this specific key. This attack requires 2^n pre-computations, 2^n plaintext/ciphertext pairs, and 2^n n-bit memory storage.

3. Exhaustive key search: This attack exhaustively examines every possible value of the 2^k keys, given plaintext/ciphertext pair; and the correct key is the one that encrypts this specific plaintext to its corresponding ciphertext. This attack has time complexity of 2^k encryptions, one plaintext/ciphertext pair, and negligible memory.

2.3 Cryptanalytic Techniques

As mentioned above, the cryptanalyst task is to recover the key or parts of the plaintext given the ciphertext under the previous attack models. Therefore, many cryptanalysis techniques were developed to evaluate the security of block ciphers. Examples of these cryptanalytic techniques include differential, linear, impossible differential and integral cryptanalysis. In what follows we give a very brief description of some of these cryptanalysis techniques.

2.3.1 Differential Cryptanalysis

Differential cryptanalysis was proposed by Biham and Shamir, who applied it to DES [28] in 1990. This attack was able to break 8rounds of DES in few minutes and is faster than the exhaustive search on 15-round of DES. Then, in 1993, the full round DES was first attacked by Biham and Shamir using the differential cryptanalysis [29, 30] in time 2^{37} encryptions using 2^{36} ciphertexts. Since then, differential cryptanalysis has became one of the most widely used analysis techniques not only on block ciphers, but also on other symmetric-key cryptographic primitives.

Differential cryptanalysis tries to exploit the advantage of having a pair of inputs having specific input difference that is highly correlated with their output difference (this output difference is obtained by encrypting the input pairs using the same key) to deduce some information about the secret key. This difference can be expressed in many ways, but the XOR difference is the most widely used notion in the differential cryptanalysis, i.e., the input difference $\varDelta X$ of the input pairs X^1 and X^2 can be expressed as follows: $\varDelta X = X^1 \oplus X^2$. This attack works under the chosen plaintext model. In iterative block ciphers, the differential on an i-round reduced cipher $E_{red} = f_i \circ \cdots \circ f_2 \circ f_1$ can be defined using the input difference $\varDelta X_1$ of the first round and the output difference $\varDelta X_{i+1}$ of the last round. While the characteristic is defined using the input difference $\varDelta X_1$, the output difference $\varDelta X_{i+1}$, and the intermediate differences $\varDelta X_j$, where $1 < j \leq i$. Therefore, the differential is a set of characteristics having the same input and output differences while the intermediate state differences can be varied.

As mentioned above, the round function has two main types of operations, namely linear and non-linear operations. The difference can be propagated deterministically with respect to the linear operations while in the non-linear operations, it propagates probabilistically. Therefore, any differential $\varDelta X_1 \xrightarrow{p} \varDelta X_{i+1}$ has a differential probability p. This differential can be used to distinguish the reduce round cipher E_{red} from a random permutation, if $p > 2^{1-n}$, for block cipher of n-bit block length. The differential probability p of a differential characteristic in an iterative block cipher can be computed by first analyzing the non-linear operations in each round. S-boxes are the most commonly used non-linear components in block ciphers. According to [99], we define the differential probability of an S-box as follows:

Definition 4 [99] Suppose T is an $m \times n$ S-box. If $\mathit{\Delta}\gamma$ is an m-bit input difference and $\mathit{\Delta}\delta$ is an n-bit output difference, then the probability of the differential $(\mathit{\Delta}\gamma, \mathit{\Delta}\delta)$ for T, written $\mathit{\Delta}\gamma \rightarrow \mathit{\Delta}\delta$, is defined to be

$$\begin{aligned} Pr_T(\mathit{\Delta}\gamma \rightarrow \mathit{\Delta}\delta) &= Pr_{P\in\{0,1\}^m}(T(P) \oplus T(P \oplus \mathit{\Delta}\gamma)) = \mathit{\Delta}\delta. \\ &= \frac{|\{X \in \{0,1\}^m | T(x) \oplus T(x \oplus \mathit{\Delta}\gamma) = \mathit{\Delta}\delta\}|}{2^m}. \end{aligned}$$

The differential probability p_j of round j, for a given key, can be computed from the multiplication of the differential probability of the active S-boxes (the active S-box is the S-box that has nonzero input difference), assuming the independence between the active S-boxes. Finally, the differential probability p of E_{red} is $p = \prod_{j=1}^{i} p_j$, assuming the independence of the differential probabilities between the rounds.

The obtained differential on E_{red} can be used to launch key recovery attack on the whole r-round cipher E. First, we gather $N = c/p$ (c is a constant) chosen plaintext pairs such that every pair has the input difference $\mathit{\Delta}X_1$ and ask the encryption oracle for their corresponding ciphertexts. Then, for each pair of the ciphertexts, the round keys $RK_r, \cdots, RK_{i+2}, RK_{i+1}$, that are involved in the computation of the output of the reduced round cipher E_{red} from the ciphertext, are guessed. Then, for each key we count the number of right pairs that have the output difference $\mathit{\Delta}X_{i+1}$. Finally, the right key is the one that has the maximal counter value.

Many approaches and tools have been developed to facilitate the cryptanalysis of block ciphers. In [109], Mouha *et al.* proposed an approach for finding the lower bound on the number of active non-linear components, in particular the S-boxes that involve a nonzero difference, and used the Maximum Differential Probability (MDP) of the S-boxes to drive an upper bound on the differential probability of the best differential characteristic. However, several block ciphers use modular addition as a source of non-linearity instead of S-boxes. Thus, for such ciphers, one needs to calculate the differential probability of modular addition to prove their security against differential cryptanalysis and its variations. In the former case of block ciphers employing, usually small-sized S-boxes, it is a simple task to calculate the differential probability of a given S-box by constructing its difference distribution table. Building a difference distribution table becomes impractical for block ciphers that employ addition modulo 2^{64} or even modulo 2^{32}. To address this problem, Mouha *et al.* [108] proposed using a graph theoretic approach to calculate the differential probability xdp^+ of addition modulo 2^n, when differences are expressed using XOR and the differential probability $adp^{\oplus}$ of XOR when differences are expressed using addition modulo 2^n with a linear time in the word length. Later on, Biryukov and Velichkov [35] developed a new framework to automatically find differential trails in ARX (Addition, Rotation and XOR) based block ciphers by constructing a partial difference distribution table containing differentials whose probabilities are greater than a fixed threshold. Currently, there is a lot of research towards finding the best differential (trail) that can be exploited to launch an attack against block ciphers as exemplified by the work done by Song *et al.* [119], Biryukov *et al.* [36], and Bannier *et al.* [19].

Differential cryptanalysis can be avoided by maximizing the number of active S-boxes (or non-linear elements employed in the cipher) in any differential trail and minimizing the maximum differential probability of the employed S-boxes in the cipher.

2.3.2 Linear Cryptanalysis

The original linear attack was developed by Matsui and Yamagishi and applied on FEAL [103] in 1992. Then, in 1993, it was applied by Matsui to break the full round DES using 2^{43} known plainttexts [102]. Its application is not limited to block ciphers only, but it is widely used against other symmetric-key primitives.

Linear cryptanalysis is a known plaintext attack which tries to find a linear relation between the input and output of a block cipher by linearizing the non-linear operations in the cipher. The most widely used linear relation for the input and output is the bitwise product operation ($\bullet$). More specifically, for the reduced round cipher $E_{red} = f_i \circ \cdots \circ f_2 \circ f_1$ with input X_1 and output X_{i+1}, the linear relation has the following form:

$$\Gamma_{X_1} \bullet X_1 \oplus \Gamma_{X_{i+1}} \bullet X_{i+1} = 0,$$

where $\Gamma_{X_1}, \Gamma_{X_{i+1}}$ indicate the input and output linear masks applied on X_1, X_{i+1}, respectively. The linear hull of this linear relation is represented by $(\Gamma_{X_1}, \Gamma_{X_{i+1}})$. Analogous to the characteristic in the differential cryptanalysis, the linear approximation can be defined by the input mask Γ_{X_1}, intermediate state masks Γ_{X_j} $(1 < j \leq i)$, and output mask $\Gamma_{X_{i+1}}$. Therefore, the linear hull is a set of linear approximations that have the same input and output masks and different intermediate state masks.

Due to the existence of the non-linear operations in any iterative block ciphers, the linear hull $\Gamma_{X_1} \xrightarrow{p} \Gamma_{X_{i+1}}$ holds with a probability p. The effectiveness of the linear hull can be determined from the bias $\varepsilon = |p - 1/2|$ of the relation, i,e., a linear approximation of a bias

$\varepsilon > 0$ can be used to distinguish the reduced round cipher E_{red} from a random permutation (its bias $\varepsilon = 0$). The correlation and capacity of a linear approximation can be defined as $c = 2 \times \varepsilon$ and c^2, respectively. In iterative block ciphers, the bias ε of a linear relation can be computed by analyzing the non-linear components, S-boxes, as in the differential cryptanalysis. First, we try to find the best linear approximation $\Gamma_{X_j} \xrightarrow{p_j} \Gamma_{X_{j+1}}$ for each round j that holds with a correlation c_j. Then, the obtained linear approximations of the i rounds are connected using the piling-up lemma [102] to obtain the reduced cipher E_{red} linear approximation $\Gamma_{X_1} \xrightarrow{p} \Gamma_{X_{i+1}}$ that holds with correlation $c = \prod_{j=1}^{i} c_j$ and bias $\varepsilon = c/2$. If we have m linear approximations, each has correlation c_h $(1 \leq h \leq m)$ with the same input and output masks, then the linear hull capacity $c^2 = \sum_{h=1}^{m} c_h^2$.

Similar to the differential cryptanalysis, the linear approximation distinguisher can be used to launch key recovery attack on the full cipher E, by gathering $N = c/\varepsilon^2$ (c is a constant) known plaintext/ciphertext pairs and guessing the round keys that are needed to compute the output parity $\Gamma_{X_{i+1}} \bullet X_{i+1}$ from the ciphertext. The correct key is the one that has a maximal number of plaintext/ciphertext pairs satisfying the linear relation.

Analogous to differential cryptanalysis, many automated tools were developed in order to facilitate the work of the cryptanalyst. The same MILP-based approach developed by Mouha *et al.* [109] can be used to obtain bounds on the minimum number of active S-boxes and thus on the maximum linear bias. Sun *et al.* utilized the MILP technique [124] to build a tool that finds the best linear trail which holds with a high probability. Later, Liu *et al.* used a SAT solver to automatically search for linear trails in ARX based block ciphers and applied this technique to the SPECK block cipher and the lightweight MAC primitive Chaskey, which is being considered for standardization by ISO/IEC and ITUT [98].

Similar to the differential cryptanalysis, this type of attacks can be avoided by maximizing the number of active S-boxes in any linear approximation and minimizing the maximum correlation of the employed S-boxes in the cipher.

2.3.3 Differential-Linear Cryptanalysis

In 1994, the differential and linear cryptanalysis were combined in one attack, namely differential-linear cryptanalysis, by Langford and Hellman [85]. This attack was proposed to reduce the data complexity that is required in the differential or linear cryptanalysis. This is achieved by the attack that was launched in [85] when it succeeds to attack 8-round of DES using 512 chosen plaintexts instead of 5000 chosen plaintexts in the traditional differential attack. The 8-round attack was launched using a 7-round distinguisher. The 7-round distinguisher is constructed using a 3-round differential that covers the first three rounds, and a 4-round linear approximation, that covers the next 4 rounds. Then, this 7-round distinguisher is used to recover 10-bit information of the last round key. The main observation in building the distinguisher is that flipping some bits in the input of the first round does not change certain bits in the output of the third round, implying that the XOR of same mask on these bits is zero. Then, exploiting these unchanged bits in a linear approximation that has an input mask containing these unchanged bits only and output mask that contains certain bits of the output of the 7^{th} round is used to build the 7-round distinguisher. Finally, the correct round key is the one that has the maximal number of ciphertext pairs that their output parities are equal. The previous distinguisher exploits a differential with probability 1. Another variation of the differential-linear cryptanalysis that exploits probabilistic differential was introduced by Biham, Dunkelman and Keller in 2002 [27]. This enhanced version was able to present the best attack on 9-round DES.

Recently this technique was successful in attacking 7-round Chaskey (Chaskey is a recent lightweight MAC that is being considered for standardization by ISO/IEC and ITU-T) in EUROCRYPT 2016 by Leurent [90].

The differential-linear attack was successful on 8-round DES because flipping bit in the input of the first round results in unchanged bits of the output of the third round. Therefore, to avoid such attacks the block cipher should reach the full diffusion after a small number of rounds.

2.3.4 Higher-Order Differential Cryptanalysis

Higher-order differential cryptanalysis, proposed by Lai [84] in 1994, is a generalization of the differential cryptanalysis. While in differential cryptanalysis the propagation of a specific difference, namely, the difference between two plaintexts, is studied in order to find highly correlated input and output differences, in higher-order differential cryptanalysis the propagation of differences between a large set of plaintexts is studied. Knudsen [80] indicated that there are block ciphers that are resistant against differential cryptanalysis, but not immune against higher-order differential cryptanalysis. As an example, for the round function $f(X, k) = (X + k)^2 \bmod q$ with block length $2 \times log_2 q$, where q is a prime number, the non-trivial one round differential probability of this function is $1/q$ while its second-order differential is constant, i.e., it holds with probability one. Later, another block cipher, proposed by Nyberg and Knudsen, which was shown to be immune against differential cryptanalysis [110], was attacked by Jakobsen and Knudsen [70] utilizing higher-order differential cryptanalysis.

In general, the higher-order differential cryptanalysis is more powerful than its special variant, i.e., the differential cryptanalysis, especially when applied to round functions employing low algebraic degree non-linear components. However, extending higher-order differential to more than 2 rounds is not as simple as it is in differential cryptanalysis.

Since the effectiveness of higher-order differential cryptanalysis depends on the algebraic degree of the non-linear components employed in the cipher, block ciphers should employ high algebraic degree non-linear elements to avoid this attack.

2.3.5 Truncated Differential Cryptanalysis

Truncated differential cryptanalysis was proposed by Knudsen in 1994 [80]. While in differential cryptanalysis, each bit of the input and output differences is specified, truncated differential cryptanalysis is more concerned whether there is a difference or not. To illustrate this point, let us take an example of a truncated differential for a 4-byte word represented as 0110, where 0 indicates that the corresponding byte is inactive, or has a zero difference, and 1 indicates that the corresponding byte is active, or has any nonzero difference. This implies that a truncated differential can be considered as multiple differentials that have zero difference in the inactive bytes and have all the possible differences in the active ones.

The truncated differential cryptanalysis was first applied to attack 6-round DES using a small number of chosen plaintexts and very modest time complexity [80]. Afterwards, Knudsen, Robshaw and Wagner, using the truncated differential cryptanalysis, proposed a set of attacks on reduced round Skipjack block cipher [81]. First, they launched an efficient key recovery attack against the first 16 rounds using practical data complexity. Second, the key of the middle 16 rounds was efficiently retrieved using two chosen plaintexts. Finally, they showed the existence of a 24-round truncated differential that holds with probability 1. Since then, truncated differential cryptanalysis has been applied widely on many block ciphers such as SAFER [139], IDEA [44], E2 [107], Camellia [89], CRYPTON [77], and KLEIN [113], to name a few. Moreover, it has been used to launch the best known attacks on 3D [83] and Midori128 [130] block ciphers.

In addition, truncated differential cryptanalysis is used in other cryptanalysis techniques such as the Meet-in-the-Middle (MitM) with differential enumeration attack that was proposed by Dunkelman, Keller, and Shamir [63]. It is also used in the MitM attacks on hash functions to launch pre-image attack and in the rebound attack to launch collision attacks.

The resistance against this attack can be achieved by employing binary permutation layer as this technique is only efficient on word oriented block ciphers.

2.3.6 Integral Cryptanalysis

Integral cryptanalysis was proposed by Daemen, Knudsen, and Rijmen in 1997 [52] to study and analyze the security of the block cipher SQUARE. Then, it was unified and formalized by Knudsen and Wagner [79]. In integral cryptanalysis, we examine the propagation of the sum (XOR) of many plaintexts, not just two as in the differential cryptanalysis. Therefore, we can consider the integral cryptanalysis as a dual technique of the differential cryptanalysis as well. Integral cryptanalysis is quite useful especially in the analysis of block ciphers with only bijective components.

In 2012, Lu *et al*. [100] proposed combining the integral attack with the MitM attack in what they called the higher-order MitM attack. They used this technique to launch 10/11/12-round attacks on Camellia-128/192/256 with FL/FL^{-1} functions and 14/16-round attacks on Camellia-192/256 without FL/FL^{-1}.

In Eurocrypt 2015, Todo proposed a new property which he named the division property [125]. The division property can be considered as a generalization of the integral property which, unlike the integral property, can exploit the algebraic degree of the block cipher. Moreover, it can be applied against bit-oriented block ciphers and block ciphers that use non-bijective components. In Crypto 2015, the division property was used to break the full-round MISTY1 block cipher based on a 6-round integral distinguisher [125] which was further enhanced in Crypto 2016 [20].

The effectiveness of this technique is based on the diffusion of the block cipher. Therefore, high diffusion rate operations should be employed in the cipher. Moreover, the division property utilizes the low algebraic degree of the non-linear elements. Hence, high algebraic degree non-linear components can provide resistance against this attack.

2.3.7 Impossible Differential Cryptanalysis

Biham, Biryukov, and Shamir noticed that not only the differential characteristic with high probability are useful. They exploit the differential characteristic of probability exactly 0, namely impossible differential cryptanalysis, to exclude the wrong keys and applied it to reduced-round Skipjack [26] where it able to attack 31 out 32 rounds.

Miss in the Middle is the general technique to construct the impossible differential distinguisher, where in the cipher $E = E_2 \circ E_1$, we try to find two differentials with probability one, the first one covers the subcipher E_1 and has the form $\mathit{\Delta}\delta \rightarrow \mathit{\Delta}\gamma$, and the second one covers the subcipher E_2^{-1}, and has the form $\mathit{\Delta}\beta \rightarrow \mathit{\Delta}\zeta$, and the intermediate differences $\mathit{\Delta}\gamma, \mathit{\Delta}\zeta$ do not match. Finally, we have the differential $\Delta\delta \to \Delta\beta$ that covers the whole cipher E and holds with zero probability.

The impossible differential distinguisher can also be used to launch attacks against block ciphers such that an i-round distinguisher can be extended to i+t rounds. Then, we choose the plaintext pairs such that their differences equal $\Delta\delta$. Then, for the corresponding ciphertext pairs we decrypt them by guessing the round keys in the t analysis rounds. Finally, we exclude the wrong keys, i.e., keys for which partial decryption has the output difference $\Delta\beta$.

The number of rounds that can be covered by the impossible differential depends on the i rounds covered by the distinguisher and the t analysis rounds. To minimize the number of rounds $i + t$ that is covered by the impossible differential attack, strong diffusion operations should be utilized in the cipher. In addition, one should also avoid the use of S-boxes with undisturbed bits.

2.3.8 Zero-Correlation Cryptanalysis

In the traditional linear cryptanalysis [103], the attacker tries to find a linear relation between an input x and an output y of an n-bit block cipher function f that has the following form:

$$\Gamma_x \bullet x \oplus \Gamma_y \bullet y = 0,$$

where $\bullet$ is a bitwise dot product operation and Γ_x (Γ_y) is the input (output) linear mask. This linear relation has a probability p, and in this type of attack it should be far from 1/2 or equivalently its correlation $C = 2 \times p - 1$ is not zero. The following lemmas are used to specify the propagation of linear masks through the different operations (XOR, branch, and S-box) that are used in the round function.

Lemma 1 *(XOR operation [39, 135]): Either the three linear masks at an XOR $\oplus$ are equal or the correlation over $\oplus$ is exactly zero.*

Lemma 2 *(Branching operation [39, 135]): Either the three linear masks at a branching point $\cdot$ sum up to 0 or the correlation over $\cdot$ is exactly zero.*

Lemma 3 *(S-box permutation [39, 135]): Over an S-box S, if the input and output masks are neither both zero nor both nonzero, the correlation over S is exactly zero.*

Later on, Bogdanov and Rijmen [39] proposed a new technique called zero-correlation cryptanalysis which, in contrast to the linear cryptanalysis, exploits linear relations with correlation exactly zero to exclude wrong keys which lead to this linear approximation. To remove the burden of the high data complexity of the zero-correlation attack and the statistical independence for multiple zero-correlation linear approximations, Bogdanov *et al.* [42] proposed the multidimensional zero-correlation attack. In this technique, we have m different linear approximations with zero-correlation, where all the $l = 2^m - 1$ non-zero linear approximations involved in the spanned linear space of these m linear approximations should have zero-correlation. The zero-correlation linear approximation over r_m rounds can act as a distinguisher, then the attacker can prepend/append additional analysis rounds. The attack proceeds by gathering N plaintext/ciphertext pairs and creating an array of counters $V[z]$, where $|z| = m$ bits, and initializing it to zero. Then, for each plaintext/ciphertext pair and key guess, the attacker computes the corresponding bits needed to apply the m linear approximations to compute z and increments the corresponding counter by one. Afterwards, the attacker computes the statistic T [42]:

$$T = \sum_{z=0}^{2^m-1} \frac{(V[z] - N2^{-m})^2}{N2^{-m}(1-2^{-m})} = \frac{N2^m}{(1-2^{-m})} \sum_{z=0}^{2^m-1} \left(\frac{V[z]}{N} - \frac{1}{2^m} \right)^2. \tag{2.1}$$

The right key has T that follows χ^2-distribution with mean $\mu_0 = l\frac{2^n-N}{2^n-1}$, and variance $\sigma_0^2 = 2l(\frac{2^n-N}{2^n-1})^2$, while the statistic for the wrong key guess follows χ^2-distribution with mean $\mu_1 = l$ and variance $\sigma_1^2 = 2l$ [42]. The number of known plaintexts required by the attack can be estimated as follows [42]:

$$N = \frac{2^n(Z_{1-\gamma} + Z_{1-\zeta})}{\sqrt{l/2} - Z_{1-\zeta}}, \tag{2.2}$$

where γ (resp. ζ) denotes the probability to incorrectly discard the right key (resp. the probability to incorrectly accept a random key as the right key) and $Z_p = \phi^{-1}(p)$ $(0 < p < 1)$, ϕ is the cumulative function of the standard normal distribution. According to the required γ and ζ probabilities, the decision threshold is set to $\tau = \mu_0 + \sigma_0 Z_{1-\gamma} = \mu_1 - \sigma_1 Z_{1-\zeta}$.

Since this technique can be seen as the dual of the impossible differential cryptanalysis, it can be avoided using countermeasures similar to those used to defend against the impossible differential cryptanalysis.

2.3.9 Basic Meet-in-the-Middle Cryptanalysis

MitM attacks can be viewed as enhanced/generalized exhaustive search techniques. These attacks try to split the block cipher into two parts; one is used in the encryption direction and the other is used in the decryption direction. Then, the attacker partially guesses key bits from both ends and propagates the knowledge of the internal state of the block cipher until the information propagated in both directions meet in the middle for matching. The key bits are considered wrong if no match is found, otherwise it is a key candidate. This technique was first introduced by Diffie and Hellman [59]. Later, it was applied to reduced-round DES by Chaum and Evertse [50].

Suppose we have an r-round cipher $E_K : \{0,1\}^n \longrightarrow \{0,1\}^n$, where K is the key that is used in the block cipher and n is the block length in bits. If this cipher can be split into two sub-ciphers F_{K_f} and G_{K_b} such that $E_K = G_{K_b} \circ F_{K_f}$ and K_f, K_b are independent subkey bits, then the attack can be launched as shown in Algorithms 1 and 2.

```
Algorithm 1: MitM attack with low space complexity
    Data: $P_i/C_i$ pairs, where $P_i/C_i$ donate the plaintext/ciphertext pairs,
          $i = 1, 2, ..., N$ and $N$ is determined by the unicity distance
    Result: Right key $k_f$ and $k_b$
    foreach $k_f \in K_f$ do
        foreach $k_b \in K_b$ do
            compute $v = F_{k_f}(P_1)$;
            compute $u = G^{-1}_{k_b}(C_1)$;
            if $v = u$ then
                Test $k_f$ and $k_b$ using $N$ plaintext/ciphertext pairs and return $k_f$ and $k_b$
                if the test succeed;
```

```
Algorithm 2: MitM attack with low time complexity
    Data: $P_i/C_i$ pairs, where $P_i/C_i$ donate the plaintext/ciphertext pairs,
          $i = 1, 2, ..., N$ and $N$ is determined by the unicity distance
    Result: Right key $k_f$ and $k_b$
    foreach $k_f \in K_f$ do
        compute $v = F_{k_f}(P_1)$ and store $v$ with the corresponding $k_f$ in table $T_1$;
    foreach $k_b \in K_b$ do
        compute $u = G^{-1}_{k_b}(C_1)$;
        if $u$ has a match in table $T_1$ then
            Test $k_f$ and $k_b$ using $N$ plaintext/ciphertext pairs and return $k_f$ and $k_b$ if
            the test succeed;
```

It is clear from Algorithms 1 and 2 that MitM attacks have the advantage of being able to offer some time-memory trade-off. The computational complexity of Algorithms 1 and 2 are $2^{|K_f|+|K_b|}$ and $2^{|K_f|} + 2^{|K_b|}$, respectively, and the memory complexity of Algorithms 1 and 2 are $\mathcal{O}(1)$ and $\mathcal{O}(2^{|K_f|})$, respectively. Algorithm 2 can be modified, by computing T_1 for the smaller key subset out of k_f, k_b and matching using the other key subset, to reduce its memory complexity to $\mathcal{O}(2^{min(|K_f|,|K_b|)})$. The data complexity of the two algorithms is the same and equals the unicity distance of the block cipher, i.e., $\lceil \frac{k}{n} \rceil$.

In spite of the fact that this technique has a low data complexity, which is the unicity distance, it got less attention than the linear and differential cryptanalysis. The reason for this is that it needs to partition the block cipher into two parts that use independent subkey bits. For modern block ciphers, this is not easy to achieve since their key schedules usually contain non-linear components such as S-boxes and/or addition modulo 2^n. Therefore, the number of rounds that can be attacked by this technique is very limited. Recently, new modifications of this basic MitM attack were proposed which allowing attackers to penetrate more rounds or even the whole cipher as we discuss in the following subsections.

The resistant against all different types of the Meet-in-the-Middle attacks can be guaranteed by employing non-linear elements in the key schedule and reach the full diffusion after small number of rounds.

2.3.10 3-Subset MitM Cryptanalysis

Bogdanov and Rechberger proposed a new variant of the basic MitM attack [43] and used it to attack the full-round KTANTAN cipher. This technique relaxes the constraint of the two keys K_f and K_b to be fully independent by splitting the key into three subsets instead of two. This approach is divided into two main stages. The *MitM stage* in which the key space is filtered depending on the matching criteria and the *key testing stage* in which the reduced key space is tested exhaustively until the right key is found.

Let $E_K : \{0,1\}^n \longrightarrow \{0,1\}^n$ be an n-bit block cipher of r rounds that uses a k-bit key K. Suppose this cipher can be split into two sub-ciphers F_{K_f} and G_{K_b} such that $E_K = G_{K_b} \circ F_{K_f}$. In this approach, K_f and K_b do not need to be fully distinct. Let $A_0 = K_f \cap K_b$ be the common key bits used in F_{K_f} and G_{K_b}, and let $A_1 = K_f \setminus A_0$ and $A_2 = K_b \setminus A_0$ denote the set of key bits that are used only in F_{K_f} and G_{K_b}, respectively. Then, the attack can be launched as shown in Algorithm 3.

The computational complexity of the two stages: MitM stage and key testing stage, can be determined as follows:

$$\overbrace{2^{|A_0|}(2^{|A_1|} + 2^{|A_2|})}^{\text{MitM stage}} + \overbrace{(2^{k-n} + 2^{k-2n} + ...)}^{\text{Key testing stage}}$$

Algorithm 3: Three-subset MitM attack [43]

```
Data: $P_i/C_i$ pairs, where $P_i/C_i$ donate the plaintext/ciphertext pairs,
      $i = 1, 2, ..., N$ and $N$ is determined by the unicity distance
Result: Right key $k_f$ and $k_b$
//MitM stage
foreach $a_0 \in A_0$ do
    foreach $a_1 \in A_1$ do
        compute $v = F_{a_0,a_1}(P_1)$;
    foreach $a_2 \in A_2$ do
        compute $u = G^{-1}_{a_0,a_2}(C_1)$;
    perform the matching between $v$ and $u$, let $|v| = |u| = n$-bit, where $|v|$
    donates the length of $v$ in bits, then we will have $2^{k-n}$ remaining candidate;
//key testing stage
the remaining candidates are exhaustively searched until the right key is found;
```

The data complexity of this approach is dominated by the data needed in order to accomplish the key testing stage, and is determined by the unicity distance of the block cipher, i.e., $\lceil \frac{k}{n} \rceil$. The memory required by this approach is used to save one of the lists A_1 or A_2. Therefore, the memory complexity is $min(2^{|A1|}, 2^{|A_2|})$. Clearly, this approach has an advantage over exhaustive search when both sets A_1 and A_2 are not empty.

2.3.11 Splice-and-Cut Cryptanalysis

Aoki and Sasaki proposed another variant of the 3-subset MitM attack to perform a preimage attack on SHA-0 and SHA-1 hash functions [16]. This approach relaxed the constraint of the MitM stage that should begin from the plaintext P and the ciphertext C to compute the intermediate state values v and u, respectively. Instead, the attack begins from an intermediate state X as shown in Figure 2.3. Then, by partially decrypting (encrypting) X to obtain plaintext (ciphertext) and using the encryption (decryption) oracle, the attacker can obtain the corresponding ciphertext (plaintext). After that, v and u can be computed from X and C (or P), respectively. This approach changes the data requirement to chosen plaintext instead of known plaintext as the previous variants. Moreover, its data complexity is higher than the 3-subset MitM variant and determined by the plaintext (ciphertext) bits that are effected by the guessed key bits, the computational complexity may be enhanced, however.

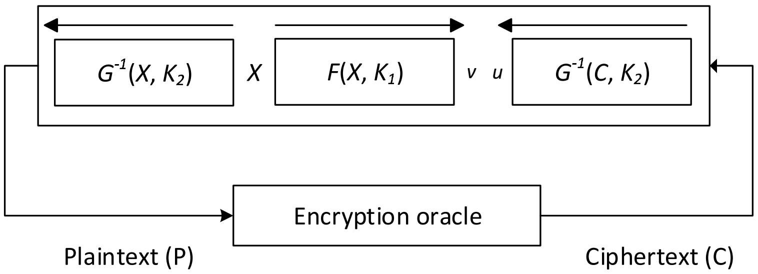


Figure 2.3: Meet-in-the-middle with a splice and cut technique

2.3.12 Multidimensional MitM and Generalized MitM Cryptanalysis

Two more variants of the MitM cryptanalysis are the Multidimensional MitM (MD-MitM) and the generalized MitM cryptanalysis technique. The first one was proposed by Zu and Gong [143]. In this approach, the block cipher is divided into multiple sub-ciphers where the number of these sub-ciphers determines the dimension of the attack, e.g., if the block cipher is divided into two sub-ciphers then the attack is called 2D-MitM attack. Then, by guessing the intermediate states, a MitM attack can be launched on each sub-cipher. Finally, the correct key is derived from the matching of the multiple MitM attacks. The basic MitM attack can be considered as a special case of MD-MitM cryptanalysis and would be denoted as 1D-MitM cryptanalysis.

The second variant, i.e., the generalized MitM cryptanalysis technique, was proposed by Tolba and Youssef [134]. While in the 3-subset MitM attack, the key space is partitioned, as mentioned above, into 3 subsets, namely, A_0, A_1 and A_2 with the two subsets A_1 and A_2 are restricted to be independent. In the generalized MitM approach, the restrictions of the 3-subset MitM attack are elevated by allowing the key space to be partitioned into $n \geq 3$ subsets and these subsets are allowed to be dependent. Using dependent subsets raises a new problem of how to efficiently compute the internal states that are dependent on two or more key subsets. To tackle this problem, the re-computation technique exploited in the biclique cryptanalysis [40], discussed latter in this chapter, is utilized.

2.3.13 Plain MitM and MitM with Efficient Enumeration Cryptanalysis

Demirci and Selçuk [55] started a new line of research on the MitM attack by launching 8-round attacks on AES-192/256 [55]. This new MitM attack, which is called plain MitM in the remaining parts of this thesis, splits the cipher into 3 subciphers such that $E = E_2 \circ E_{mid} \circ E_1$, where E_{mid} used as a distinguisher that its property is evaluated offline. Then, the keys in the outer rounds E_1, E_2 are guessed to verify the distinguishing property. Finally, the key is considered wrong if they do not satisfy the distinguishing property.

The distinguishing property is a truncated differential where its input takes a set of possible values and its output is a parameterized function of the input. The values of the output corresponding to the input form an ordered sequence that is used as our property to identify the right key guess. All the ordered sequences resulting from all the possible combinations of the parameters are stored in a precomputation table. The δ*-set* and b-δ*-set* concept [52, 67], as captured by Definition 5 and 6, respectively, are used to build the distinguishing property.

Definition 5 *(δ-set, [52]). A δ-set for nibble(byte)-oriented cipher is a set of 16(256) state values that are all different in one nibble(byte) (the active nibble(byte)) and are all equal in the remaining nibbles(bytes) (the inactive nibbles(bytes)).*

Definition 6 *(b-δ-set, [67]). A b-δ-set is a set of 2^b state values that are all different in b state bits (the active bits) and are all equal in the remaining state bits (the inactive bits).*

The distinguishing property that is used to launch the attack on AES [55] is that each output byte after 4 rounds of AES can be considered as a parametrized function of the input, hence it depends on 25 byte parameters when the input has only one active byte [55]. Then, the number of parameters were reduced to 24 byte parameters [56] by considering the difference of the output byte instead of its value, as one of the parameters is the key and it is fixed for all the functions. Even after the reduction in the number of parameters, this technique requires high memory storage to store the precomputation table that contains all the possible ordered sequences. Therefore, this attack was only applied on AES-192/256.

To reduce the high memory requirements of the MitM attack, new ideas were proposed by Dunkelman, Keller, and Shamir at ASIACRYPT [63]. The first idea is to store the multiset, as captured by Definition 7, instead of the ordered sequence. More specifically, storing unordered sequence associated with its multiplicity, and using this idea, the memory complexity was reduced by a factor of 4. The second idea, namely differential enumeration, allows us to reduce the number of parameters from 24 to 16 by using truncated differential with low probability. Utilizing these two ideas reduces the memory requirements from 2^{192} to 2^{128}, but still not enough to attack AES-128. The consequence of using such low probability truncated differential is that the data that are required to verify the distinguisher were increased.

Definition 7 *(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 as many as $\binom{2^8+2^8-1}{2^8} \approx 2^{506.17}$ different values.*

Later on, AES-128 was attacked by Derbez, Fouque and Jean [58] utilizing ideas from the rebound attack [104] that are based on the S-box differential property as captured by Proposition 1. They showed that the number of parameters can be reduced from 16 to 10 byte parameters using the new idea that called efficient enumeration. Consequently, they presented the most efficient attack on 7-round AES-128. Moreover, they presented a 9-round attack on AES-256 using a 5-round distinguisher.

Proposition 1 *(Differential Property of the S-box) Given two nonzero differences Δi and Δo in $\mathbb{F}16$ or $\mathbb{F}256$, the equation: $S(x) + S(x + \Delta i) = \Delta o$ has one solution on average. This property also applies to S^{-1}.*

Finally, further reduction of the memory requirements was achieved by proposing a new idea called key dependent sieving [91]. Using this idea Li, Jia and Wang presented 9-round attack on AES-192 using a 5-round distinguisher. Then, Li and Jin [93] presented the first 6-round distinguisher on AES-256 that is used to present the first 10-round attack on AES-256.

Since then, MitM has become one of the most powerful cryptanalysis techniques against block ciphers. It was applied to SPN block ciphers such as Hierocrypt-3 [6], Hierocrypt-L1[8], ARIA [11], and Kalyna [12]; and FN ciphers such as TWINE [34], Khudra [127], Piccolo [129], CLEFIA [92], and Camellia [92].

2.3.14 Biclique Cryptanalysis

The biclique cryptanalysis was proposed by Bogdanov *et al.* [40] where they succeeded in theoretically attacking the full-round AES block cipher. This technique consists of two parts: building a biclique for some rounds of the block cipher at its beginning or end and applying the MitM with re-computation technique on the remaining part of the block cipher. The technique was inspired by the splice-and-cut technique and its general structure is depicted in Figure 2.4. The attack works in the single-key setting and has two main parameters: the *length* which denotes the number of rounds covered by the biclique and the *dimension* which denotes the length of the biclique elements.

Here, we give a brief definition of biclques as applied to this class of attacks. Suppose we have a sub-cipher F_K which maps a plaintext P to intermediate state S in which we have 2^d plaintexts P_i and 2^d intermediate states S_j connected by 2^{2d} keys $K[i,j]$ as shown in Figure 2.5. A d-dimensional biclique is the tuple $(P_i, S_j, K[i,j])$ where the following relation holds:

$$S_j = F_{K[i,j]}(P_i); \forall i, j \in 0, ..., 2^{d-1}$$

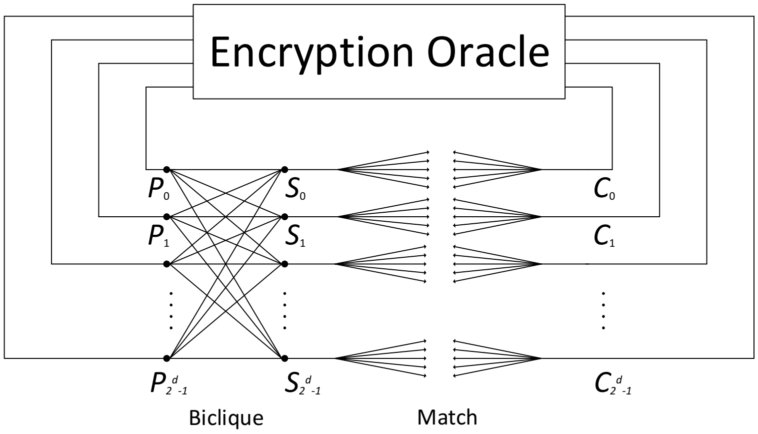


Figure 2.4: General structure of the biclique cryptanalysis

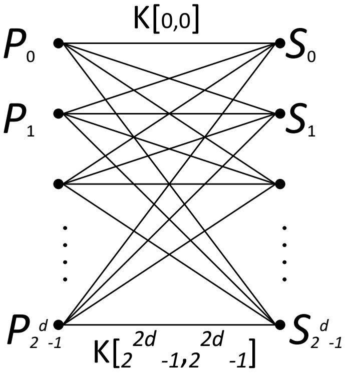


Figure 2.5: d-dimensional biclique

This technique succeeds with probability 1 since all the key space is exhaustively tested. The time complexity of this technique is given by:

$$2^{n-2d}(C_{biclique} + C_{precomp} + C_{recomp} + C_{falsepos})$$

where $C_{biclique}$ is the time needed to construct the biclique, $C_{precomp}$ is the time required to compute the elements affected by K_1 or K_2, C_{recomp} is the time required to compute the elements affected by both K_1 and K_2, and $C_{falsepos}$ is the time needed to check the remaining candidates after the matching.

2.3.15 Unbalanced Biclique Cryptanalysis

The original biclique attack succeeded in attacking the full-round versions of AES, i.e., AES-128/192/256 [40]. However, its data complexity is very high not only for AES but also when the attack is applied on other lightweight block ciphers such as LBlock [137] and TWINE [49]. To address this issue, some ideas were proposed [48, 10, 38]. All these ideas

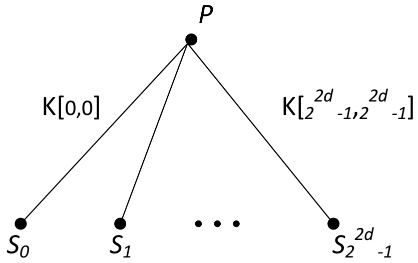


Figure 2.6: Star biclique

focused on changing the structure of the biclique from balanced where the two sets P_i and S_j of the biclique have the same cardinality 2^d, to unbalanced biclique where the two sets have different cardinality. An example of the unbalanced biclique is the star biclique in which one of the sets has cardinality one. The star biclique was first proposed by Canteaut *et al.* [48]. They used the star biclique with the MitM attack to reduce the data complexity of the MitM stage to a single plaintext/ciphertext pair [48].

The star biclique can be placed at the beginning or the end of the block cipher. For simplicity and without loss of generality, we place it at the beginning of the block cipher. As seen in Figure 2.6, we have only one plaintext P mapped to intermediate states S_l, where $l = 0, 1, ..., 2^{2d} - 1$ and each l is equivalent to a unique i, j pair, where i, j take values from 0 to $2^d - 1$. Each $S_l \equiv S_{i,j}$ is obtained by partially encrypting P using key $K[i, j]$. Such structure is called a star biclique with dimension d.

2.3.16 Invariant Subspace Cryptanalysis

Invariant subspace attack was first proposed by Leander *et al.* [86] at Crypto 2011 to cryptanalyze the PRINTcipher block cipher. For simplicity and without loss of generality, assume that we have an n-bit block cipher whose round function E_K consists of a key addition followed by an SP-layer E, such that $E_k(x) = E(x + k)$; and E has the following property:

$$E(U + c) = U + d,$$

where U is a subspace such that $U \subseteq \mathbb{F}_2^n$ and $c, d \in \mathbb{F}_2^n$ are two constants. Then, for the round keys of the form $k = u + c + d$ and $u \in U$, the following holds:

$$E_k(U + d) = E((U + d) + (u + c + d)) = E(U + c) = U + d,$$

i.e., the round function maps the affine subspace $U + d$ onto itself. Consequently, if all the round keys belong to the subspace $U + c + d$, then there exists a very efficient distinguisher for a fraction of keys over any number of rounds. Consequently, for PRINTcipher-48 (resp.

PRINTcipher-96), there exist 2^{52} weak keys out of 2^{80} (resp. 2^{102} out of 2^{160} weak keys). This attack requires 5 chosen plaintexts if the key belongs to the weak keys class which is considered as a low data complexity attack.

Afterwards, a complete study of the attack against PRINTcipher was done by Bulygin *et al.* [45]. Later, in Eurocrypt 2015, Leander *et al.* [87] proposed a generic algorithm to detect the invariant subspaces and applied it on iSCREAM (one of the CAESAR competition candidates), Robin (LS-design), and Zorro (a lightweight block cipher). Then, it was used by Guo *et al.* [66] to present a full-round attack against Midori64. Finally, Grassi *et al.* [65] proposed the subspace trail cryptanalysis, which can be considered as a generalization to the invariant subspace attack in which they no longer rely on specific choices of round constants or subkeys, and applied this technique on AES to present a 5-round distinguisher in the single-key setting.

This attack can be avoided by carefully choosing the round constants, avoiding S-boxes that map a subspace to itself, using non-linear elements in the key schedule, and using operations with high diffusion rates.

Chapter 3

MitM Attacks on Khudra and Piccolo

3.1 Introduction

Recently, there has been a huge demand for deploying resource-constrained devices such as RFID tags and wireless sensor nodes. To provide cryptographic security to such resource-constrained devices, new block ciphers utilizing simple round functions, and modest or even no key schedules are developed. As such, the design and analysis of hardware-oriented lightweight block ciphers have become a very pressing task for the cryptographic community. HIGHT [68], mCrypton [94], DESL/DESXL [88], PRESENT [41], KATAN/KTANTAN [46], PRINTcipher [78], and Piccolo [116] are just few examples of such lightweight block ciphers that are designed to be efficiently deployed on resource-constrained devices.

Khudra [82] and Piccolo [116] are hardware-oriented lightweight block ciphers.Both ciphers operate on a 64-bit state with different key sizes. Their structure inherit the Generalized Feistel Network (GFN) construction and has 4 branches, each of 16-bit length. They have been analyzed extensively as exemplified in [101, 76, 142, 74, 136, 75, 118].

In this chapter, we investigate the security of Khudra and Piccolo against the MitM attack based on the Demirci and Selçuk plain MitM approach. As mentioned in chapter 2, this attack is based on finding a distinguisher that is evaluated offline independently of the key. Then in an online phase, some rounds are appended before and after the distinguisher and the correct key candidates for these rounds are checked whether they verify the distinguisher property or not.

The rest of the chapter is organized as follows. Our attacks against 13 and 14 rounds of Khudra are presented in section 3.2. In section 3.3, we provide the details of Our attacks on 14 rounds of Piccolo-80 and 16, 17 rounds of Piccolo-128. Finally, the chapter is concluded in section 3.4.

3.2 Plain MitM Attack on Khudra

In this section, we present plain MitM attacks on 13 and 14 rounds of Khudra. In the attack on 13 rounds, we first construct a 6-round distinguisher, append three rounds at the top and four rounds at the bottom. To attack 14 rounds, the same distinguisher would require the whole key to be guessed, therefore we construct a different 6-round distinguisher, and append three rounds at the top and five rounds at the bottom. The time complexities of these attacks are $2^{66.11}$ to attack 13 rounds and $2^{66.19}$ to attack 14 rounds, respectively. Both attacks require the same data and memory complexities of 2^{51} chosen plaintext and $2^{64.8}$ 64-bit blocks.

3.2.1 Specifications of Khudra

Khudra is an iterated lightweight block cipher that operates on 64-bit blocks using an 80-bit key and employs a Generalized Feistel Structure (GFS). It has four branches of 16-bit each, i.e., the state is divided into four words and each word is 16-bit long. The cipher iterates over 18 rounds where in every round, an unkeyed 16×16-bit F-function is applied on two words. This unkeyed F-function, designed to be efficient when deploying Khudra on FPGAs, uses a 6-round GFS as depicted in the right side of Figure 3.1. Each round of these 6-round GFS has two 4×4-bit S-boxes identical to the S-box used in PRESENT [41]. After applying the F-functions of round i, two 16-bit round keys RK_{2i} and RK_{2i+1} are XORed to the state along with the other two words to generate the two new words of round $i+1$ for $i = 0, 1, \cdots, 17$. Additionally, two pre-whitening keys WK_0 and WK_1 are XORed with the plaintext before the first round and two other post-whitening keys WK_2 and WK_3 are XORed with the internal state after the last round and before generating the ciphertext.

The key schedule of Khudra takes an 80-bit master key K and splits it into five keys k_i of 16-bit each where $K = k_0||k_1||k_2||k_3||k_4$. Then, it generates 16-bit 36 round keys $RK_i, 0 \leqslant i < 36$, two per round, and four 16-bit whitening keys $WK_i, 0 \leqslant i < 4$, as shown in Algorithm 4.

Algorithm 4: The Key Schedule employed in Khudra [82]

```
Data: Key Scheduling($k_0, k_1, k_2, k_3, k_4$)
Result: $WK_i, 0 \leqslant i < 4$ and $RK_i, 0 \leqslant i < 36$
$WK_0 \leftarrow k_0, WK_1 \leftarrow k_1, WK_2 \leftarrow k_3, WK_3 \leftarrow k_4$;
for $i \leftarrow 0$ to 35 do
    $RC_i \leftarrow 0||i_{(6)}||00||i_{(6)}||0$;
    $RK_i \leftarrow k_{i\,mod\,5} \oplus RC_i$;
```

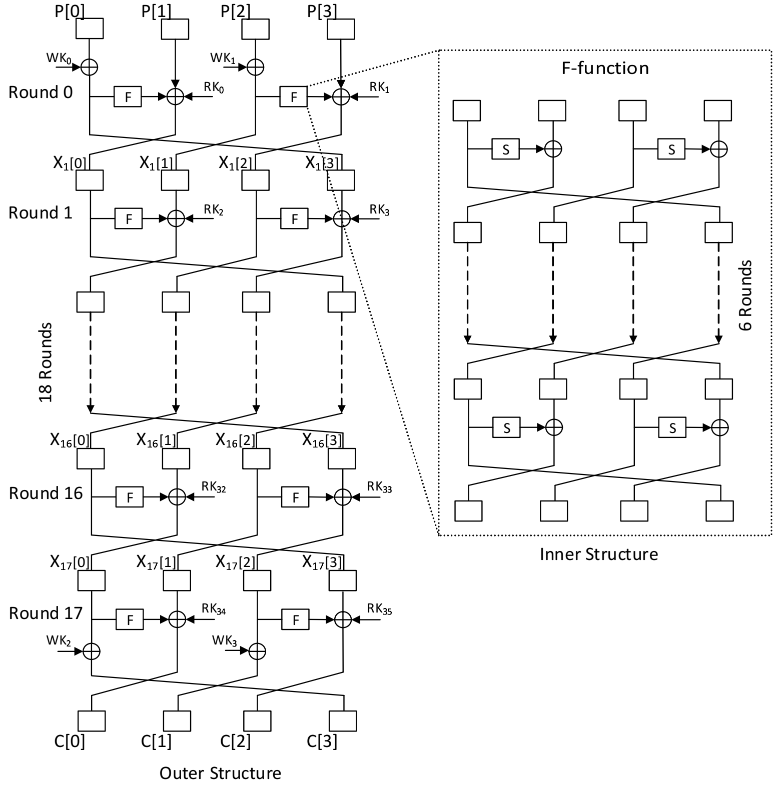


Figure 3.1: Structure of Khudra

The following notations will be used throughout the rest of this section:

- K: The master key.

- k_i: The i^{th} 16-bit of K, where $0 \leq i < 5$.

- RK_i: The 16-bit key used in round $\lfloor i/2 \rfloor$.

- WK_i: The 16-bit whitening key, where $0 \leq i < 4$.

- X_i: The 64-bit input to round i, where $0 \leq i \leq 18$, X_0 is the plaintext P and X_{18} is the ciphertext C.

- $X_i[l]$: The l^{th} 16-bit word of X_i, where $0 \leq l < 4$.

- $\Delta X_i, \Delta X_i[l]$: The difference at state X_i and word $X_i[l]$, respectively.

- X_i^j: The j^{th} state of the 64-bit input to round i.

- $X_i^j[l]$: The l^{th} 16-bit word of the j^{th} state of the 64-bit input to round i.

We measure the memory complexity of our attacks in number of 64-bit Khudra blocks and the time complexity in terms of the equivalent number of round-reduced Khudra encryptions.

3.2.2 A MitM Attack on 13-Round Khudra

A *b-δ-set* (see Definition 6) is employed in our MitM attack where we set $b = 3$, i.e., 3 active bits. b is chosen in order to reduce the memory and data requirements of the attack without increasing its time complexity. In our 13-round attack, the active word is $P[1]$, i.e., the second word. The 3 active bits can take any position in this 16-bit word. Such 3-*δ-set* enables us to build a 6-round distinguisher, as depicted in Figure 3.2, by the following proposition:

Proposition 2 *Consider the encryption of* 3-*δ-set* $\{P^0, P^1, ..., P^7\}$ *through six rounds of Khudra. The ordered sequence* $[X_6^0[3] \oplus X_6^1[3], X_6^0[3] \oplus X_6^2[3], ..., X_6^0[3] \oplus X_6^7[3]]$ *is fully determined by the following 4 16-bit parameters,* $X_1^0[0]$, $X_2^0[0]$, $X_3^0[0]$ *and* $X_4^0[0]$.

The above proposition means that we have $2^{4 \times 16} = 2^{64}$ ordered sequences out of the $2^{(2^3-1) \times 16} = 2^{112}$ theoretically possible ones.

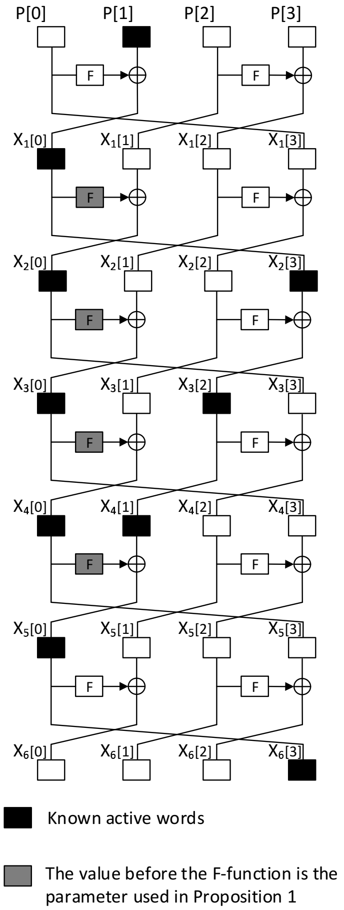


Figure 3.2: 6-round distinguisher to attack 13-round Khudra

Proof. The knowledge of the 3-*δ-set* $= \{P^0, P^1, \cdots, P^7\}$ allows us to determine $[P^0 \oplus P^1, P^0 \oplus P^2, \cdots, P^0 \oplus P^7]$. In what follows we show how the knowledge of the 4 16-bit parameters mentioned in Proposition 2 is enough to compute the ordered sequence of the differences at $X_6[3]$. As there is no F-function involved in the first round, the difference $\Delta P[1]$ is propagated through the first round as is. The knowledge of $X_1^0[0]$ enables us to bypass the F-function of the second round to compute $\Delta X_2[0]$. Then, the knowledge of $X_2^0[0]$ enables us to bypass the F-function of the third round to compute $\Delta X_3[0]$ and the previous steps are repeated until we compute $\Delta X_6[3]$. It is to be noted that after the third (resp. fourth) round, $X_3[3]$ (resp. $X_4[3]$) should have non-zero difference because $X_2[0]$ (resp. $X_3[0]$) has non-zero difference. However, these differences are omitted from Figure 3.2 since they do not impact the ordered sequence at $X_6[3]$.

The previous proposition is utilized to attack 13-round Khudra by appending 3 rounds on top of it and 4 rounds below it, as illustrated in Figure 3.3. The attack has two phases and proceeds as follows:

Offline Phase. Build the distinguisher property by determining all the 2^{64} ordered sequences as illustrated in Proposition 2 and save them in a hash table H.
Online Phase. As illustrated in Figure 3.3, the online phase advances as follows:

1. A plaintext P^0 is chosen to act as a reference to all the differences in the 3-δ-set.

2. The 3-δ-set P^0, P^1, $\cdots$, P^7 is determined by guessing the state variables $X_1^0[3]$, $X_1^0[1]$, $X_1^0[0]$, $X_2^0[2]$ to decrypt the 3-δ-set differences $[X_3^0[1] \oplus X_3^1[1], X_3^0[1] \oplus X_3^2[1], \cdots, X_3^0[1] \oplus X_3^7[1]]$.

3. The corresponding ciphertexts C^0, C^1, $\cdots$,C^7 are requested.

4. The differences in $[X_9^0[3] \oplus X_9^1[3], X_9^0[3] \oplus X_9^2[3], \cdots, X_9^0[3] \oplus X_9^7[3]]$ are determined by guessing the state variables $X_9^0[2]$, $X_{10}^0[0]$, $X_{11}^0[0]$, $X_{11}^0[2]$, $X_{12}^0[0]$, $X_{12}^0[2]$ that are required to decrypt the ciphertext differences $[C^0 \oplus C^1, C^0 \oplus C^2, \cdots, C^0 \oplus C^7]$.

5. The guessed state variables are filtered by checking if the computed ordered sequence exists in H or not.

Steps 2 and 4 require the guessing of 10 words and the attack time complexity would then exceed the exhaustive key search. Therefore, we investigate the key schedule aiming to find relations between the round keys and thus reduce the number of guessed words. Indeed, we find that by guessing k_0, k_1, k_3, and with the knowledge of P^0, we can compute $X_1^0[3]$, $X_1^0[1]$, $X_1^0[0]$, $X_2^0[2]$ and by guessing k_0, k_3, k_4, and with the knowledge of C^0, C^1, $\cdots$, C^7 and $[C^0 \oplus C^1, C^0 \oplus C^2, \cdots, C^0 \oplus C^7]$, we can compute $X_9^0[2]$, $X_{10}^0[0]$, $X_{11}^0[0]$, $X_{11}^0[2]$, $X_{12}^0[0]$, $X_{12}^0[2]$. Therefore, instead of guessing 10 words, only 4 key words k_0, k_1, k_3, k_4 are to be guessed.

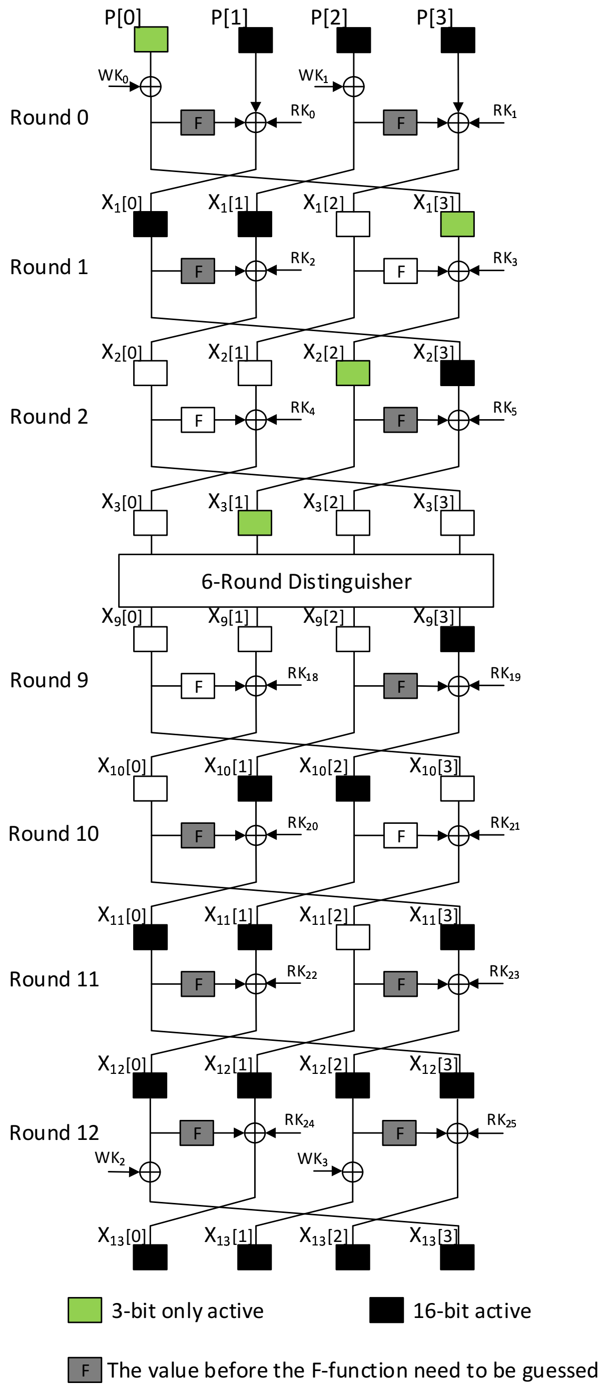


Figure 3.3: 13-round attack on Khudra

The probability of a wrong key resulting in an ordered sequence in H is $2^{64-(7\times 16)} = 2^{-48}$. As we have 2^{64} key guesses, we expect that only $2^{64-48} = 2^{16}$ keys will remain. Hence, we guess k_2 to fully recover the master key and test it using two plaintext/ciphertext pairs.

Attack Complexity. The memory complexity of the attack is determined by the memory required to store the hash table H in the offline phase. This table has 2^{64} entries where each entry contains seven 16-bit words, i.e., 112 bits. Therefore, the memory complexity is given by $2^{64} \times 112/64 = 2^{64.8}$ 64-bit blocks. The data complexity is determined from step 2. As shown in Figure 3.3, after the decryption of step 2, three words are fully active, i.e., they assume all the 2^{16} possible values while the fourth word has only three active bits, i.e., assumes 2^3 possible values only in correspondence to the 3-δ-set. Therefore, the data complexity of the attack is upper bounded by 2^{51} chosen plaintext. The time complexity of the offline phase is determined by the time required to build the hash table H and is estimated to be $2^{64} \times 8 \times 4/(2 \times 13) = 2^{64.3}$. The complexity of the online phase includes the time required to filter the key space and is estimated to be $2^{64} \times 8 \times (4+6)/(2 \times 13) = 2^{65.62}$. It also includes the time to exhaustively search through the remaining key candidates along with the guess of k_2 using two plaintext/ciphertext pairs and is estimated to be $2 \times 2^{(64-48)} \times 2^{16} = 2^{33}$. Therefore, the overall time complexity of the attack is estimated to be $2^{64.3} + 2^{65.62} + 2^{33} \approx 2^{66.11}$ 13-round Khudra encryptions.

3.2.3 A MitM Attack on 14-Round Khudra

Reusing the same distinguisher to extend our attack by one round requires guessing the 5 words of the key. Therefore, we construct another distinguisher, depicted in Figure 3.4, to attack 14-round reduced Khudra without the post-whitening keys. The active word in this new distinguisher is $P[3]$. It is built according to Proposition 3 below and, as in the previous attack, b is set to 3.

Proposition 3 *Consider the encryption of* 3-δ-*set* $\{P^0, P^1, \cdots, P^7\}$ *through six rounds of Khudra. The ordered sequence* $[X_6^0[1] \oplus X_6^1[1], X_6^0[1] \oplus X_6^2[1], \cdots, X_6^0\ [1] \oplus X_6^7[1]]$ *is fully determined by the following 4 16-bit parameters* $X_1^0[2]$, $X_2^0[2]$, $X_3^0[2]$ *and* $X_4^0[2]$.

By appending three rounds on top of this new distinguisher and five rounds beneath it, we are able to attack 14-round Khudra. The attack proceeds as the previous one, as illustrated in Figure 3.5, with the exception that the active word is $X_3[3]$ rather than $X_3[1]$ in the 13-round attack and the ordered sequence is calculated at $X_9[1]$ instead of $X_9[3]$. Guessing k_0, k_1, k_2 with the knowledge of P^0 enables us to compute the state variables needed to determine the 3-δ-set. In order to determine the ordered sequence, we need to guess k_0, k_1,

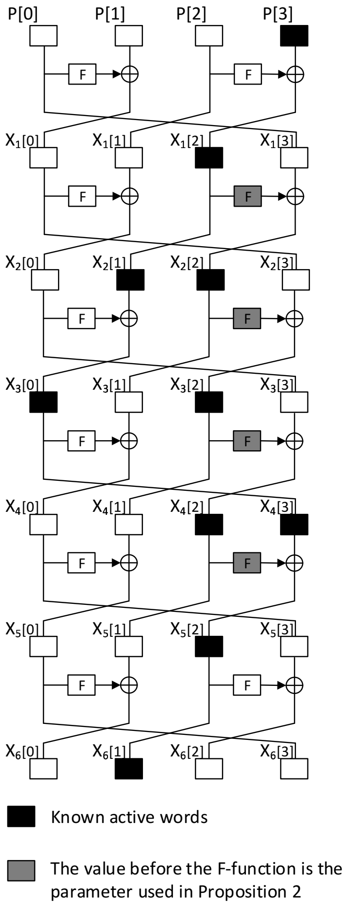


Figure 3.4: 6-round distinguisher to attack 14-round Khudra

k_2, k_4. Therefore, guessing the four key words, k_0, k_1, k_2, k_4 allows us to mount an attack on 14-round Khudra.

Attack Complexity. The memory and data complexities of this attack are similar to the previous one, i.e., $2^{64.8}$ 64-bit blocks and 2^{51} chosen plaintext, respectively. The time complexity is $2^{64} \times 8 \times 4/(2 \times 14) + 2^{64} \times 8 \times (4 + 8)/(2 \times 14) + 2 \times 2^{(64-48)} \times 2^{16} = 2^{64.19} + 2^{65.78} + 2^{33} \approx 2^{66.19}$ 14-round Khudra encryptions.

3.3 Plain MitM Attack on Piccolo

In this section, we present plain MitM attacks on 14-round reduced Piccolo-80 and 16, 17-round reduced Piccolo-128. In the attack on Piccolo-80, we first construct a 5-round distinguisher then append 4 rounds at its top and 5 rounds at its bottom. The time, data, and memory complexities of the 14-round attack on Piccolo-80 are $2^{75.39}$ encryptions, 2^{48} chosen plaintexts, and $2^{73.49}$ 64-bit blocks, respectively. To attack 16-round reduced Piccolo-128, we build a 7-round distinguisher then append 3 rounds at its top and 6 rounds at its bottom.

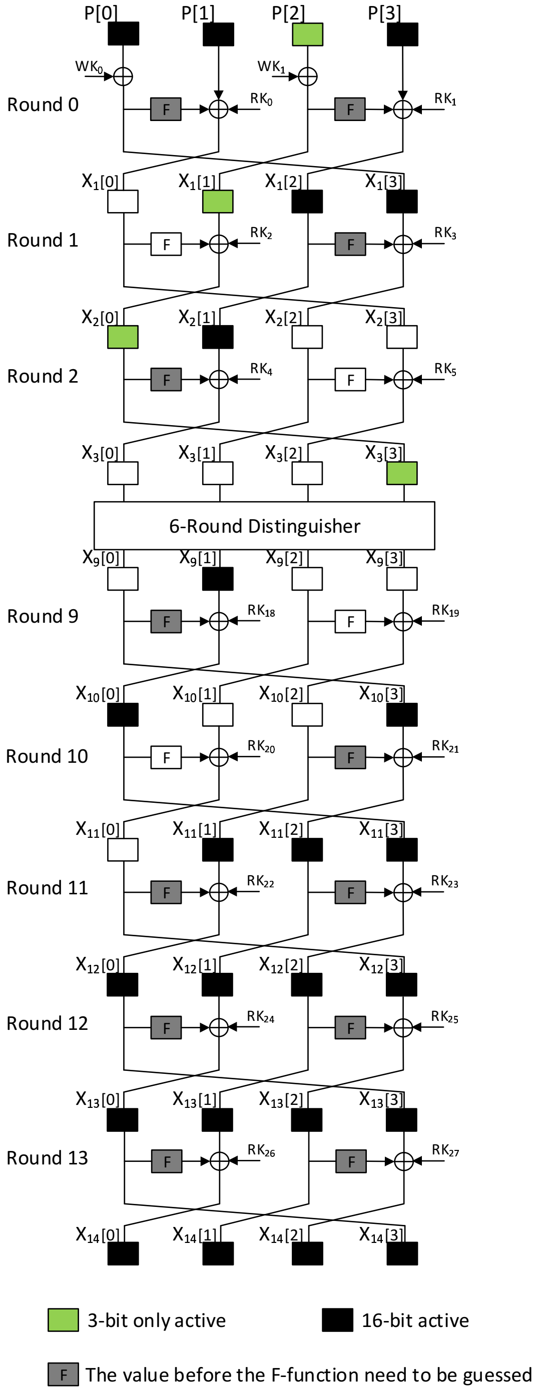


Figure 3.5: 14-round attack on Khudra

Table 3.1: Summary of the cryptanalysis results on Piccolo-80 (ID: Impossible Differential, RK: Related-Key Setting Attack, SK: Single-Key Setting Attack, Pre: Pre-whitening Key, Post: Post-whitening Key, CC: Chosen Ciphertext, CP: Chosen Plaintext, †: Requires the full codebook or more)

Attack	Setting	# Rounds	Pre/Post	Time	Data	Memory	Reference
ID	RK	14	None	$2^{68.19}$	$2^{68.19\dagger}$	N.A.	[106]
MitM	SK	14	None	2^{73}	$2^{64\dagger}$	2^{5}	[69]
ID	SK	12	Pre	$2^{55.18}$	$2^{36.34}$ CC	2^{63}	[17]
ID	SK	13	None	$2^{69.7}$	$2^{43.25}$ CP	2^{62}	[17]
MitM	SK	14	None	$2^{75.39}$	2^{48} CP	$2^{73.49}$	Sec. 3.3.2

Table 3.2: Summary of the cryptanalysis results on Piccolo-128 (ID: Impossible Differential, RK: Related-Key Setting Attack, SK: Single-Key Setting Attack, Pre: Pre-whitening Key, Post: Post-whitening Key, CP: Chosen Plaintext, †: Requires the full codebook or more)

Attack	Setting	# Rounds	Pre/Post	Time	Data	Memory	Reference
ID	RK	21	None	$2^{117.77}$	$2^{117.77\dagger}$	N.A.	[106]
MitM	SK	21	None	2^{121}	$2^{64\dagger}$	2^{6}	[69]
ID	SK	15	Post	$2^{125.4}$	$2^{58.7}$ CP	2^{61}	[17]
MitM	SK	16	Post	2^{123}	2^{48} CP	$2^{113.49}$	Sec. 3.3.3
MitM	SK	17	Post	$2^{126.87}$	2^{48} CP	$2^{125.99}$	Sec. 3.3.4

Extending the attack by one round using that 7-round distinguisher would require the whole key to be guessed. Hence, we construct a 6-round distinguisher, append 4 rounds at its top and 7 rounds at its bottom. The data complexity of both attacks on 16 and 17-round reduced Piccolo-128 is 2^{48} chosen plaintexts. The time, and memory complexities of the 16-round attack on Piccolo-128 are 2^{123} encryptions, and $2^{113.49}$ 64-bit blocks, respectively. The time, and memory complexities of the 17-round attack on Piccolo-128 are $2^{126.87}$ encryptions, and $2^{125.99}$ 64-bit blocks, respectively. Table 3.1 and 3.2 summarize our results and the previous results on Piccolo-80 and Piccolo-128, respectively.

3.3.1 Specifications of Piccolo

The following notations are used throughout the rest of this section:

- $a_{(b)}$: A word a of length b bits.
- $a||b$: Concatenation of the two words a and b.

- a^t: Transposition of the vector or the matrix a.

- a_b: Representation of the word a in base b.

- K: The master key.

- k_i: The i^{th} 16-bit of K from left, where $0 \leq i < 5$ in Piccolo-80 and $0 \leq i < 8$ in Piccolo-128.

- rk_i: The 16-bit key used in round $\lfloor i/2 \rfloor$.

- wk_i: The 16-bit whitening key, where $0 \leq i < 4$.

- X_i: The 64-bit input to round i, where $0 \leq i \leq 26$ in Piccolo-80 and $0 \leq i \leq 32$ in Piccolo-128, X_0 is the plaintext P and X_{26} or X_{32} is the ciphertext C in Piccolo-80 and Piccolo-128, respectively.

- $X_i[j]$: The j^{th} nibble of X_i, where $0 \leq j < 16$.

- $X_i[j:l]$: The nibbles from j to l of X_i, where $j < l$.

- $X_i[j,l]$: The nibbles j and l of X_i.

- $\Delta X_i, \Delta X_i[j]$: The difference at state X_i and nibble $X_i[j]$, respectively.

- X_i^j: The j^{th} state of the 64-bit input to round i.

There are two versions of Piccolo, depending on the key size, Piccolo-80 for 80-bit keys and Piccolo-128 for 128-bit keys. There are two differences between Piccolo-80 and Piccolo-128, the first is the number of rounds. Piccolo-80 iterates over 25 rounds, while Piccolo-128 runs 31 rounds. Piccolo's design employs a Generalized Feistel Network (GFN) structure and its internal state is divided into 4 words each of 16-bit length, i.e., we have 4 branches as shown in Figure 3.6. Therefore, each round has two Feistel Networks (FN). Each FN has two operations: an F-function (F) and an Add key (AK). The F-function is an unkeyed 16×16-bit function and is applied to the first branch of the FN and, as depicted in the right part of Figure 3.6, consists of three transformations [116]:

1. First S-box layer: A nonlinear layer that applies the same 4×4-bit bijective S-box S to the 16-bit $X_{(16)} = x_{0(4)}||x_{1(4)}||x_{2(4)}||x_{3(4)}$ data of the first branch of the FN as follows:

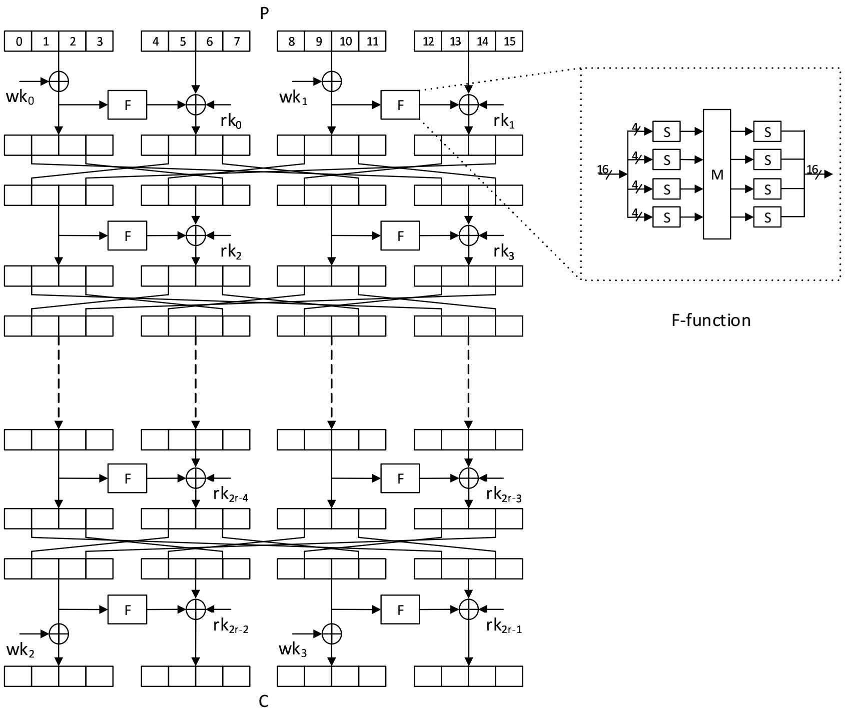
$$(x_{0(4)}, x_{1(4)}, x_{2(4)}, x_{3(4)}) \leftarrow (S(x_{0(4)}), S(x_{1(4)}), S(x_{2(4)}), S(x_{3(4)}))$$

2. Diffusion layer: The internal state is multiplied by a matrix M, where the multiplication is performed over $GF(2^4)$ defined by the irreducible polynomial $x^4 + x + 1$. Hence, the output of the first S-box layer is updated as follows:

$$(x_{0(4)}, x_{1(4)}, x_{2(4)}, x_{3(4)})^t \leftarrow M.(x_{0(4)}, x_{1(4)}, x_{2(4)}, x_{3(4)})^t,$$

3. Second S-box layer: It resembles the first S-box layer but applied to the output of the diffusion layer.

Each round of Piccolo contains two round keys used in the two FNs. Moreover, there are two pre-whitening keys wk_0, wk_1 that are xored with the internal state before the first round and two post-whitening keys wk_2, wk_3 that are xored with the internal state after the last round. After applying the two FN operations in each round, a permutation is performed on the byte level, as shown in Figure 3.6.



Encryption function

F-function

Figure 3.6: Structure of Piccolo

The key schedule takes an 80-bit master key K in Piccolo-80 such that $K = k_0||k_1||k_2||k_3||k_4$ or an 128-bit master key K in Piccolo-128 such that $K = k_0||k_1||k_2||k_3||k_4||k_5||k_6||k_7$ and generates the 4 16-bit whitening keys wk_i, $0 \leq i < 4$ and 50 16-bit round keys in Piccolo-80, as per Algorithm 5 or 62 16-bit round keys in Piccolo-128, as per Algorithm 6.

Algorithm 5: The Key Schedule employed in Piccolo-80 [116]

```
Data: Key Scheduling($k_0, k_1, k_2, k_3, k_4$)
Result: $wk_i, 0 \leqslant i < 4$ and $rk_i, 0 \leqslant i < 50$
$wk_0 \leftarrow k_0^L||k_1^R, wk_1 \leftarrow k_1^L||k_0^R, wk_2 \leftarrow k_4^L||k_3^R, wk_3 \leftarrow k_3^L||k_4^R$;
for $i \leftarrow 0$ to 24 do
    $(rk_{2i}, rk_{2i+1}) \leftarrow (con_{2i}^{80}, con_{2i+1}^{80}) \oplus \begin{cases} (k_2, k_3) & \text{if } i \ mod \ 5 = 0 \ or \ 2 \\ (k_0, k_1) & \text{if } i \ mod \ 5 = 1 \ or \ 4 \\ (k_4, k_4) & \text{if } i \ mod \ 5 = 3, \end{cases}$
```

Algorithm 6: The Key Schedule employed in Piccolo-128 [116]

```
Data: Key Scheduling($k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7$)
Result: $wk_i, 0 \leqslant i < 4$ and $rk_i, 0 \leqslant i < 62$
$wk_0 \leftarrow k_0^L||k_1^R, wk_1 \leftarrow k_1^L||k_0^R, wk_2 \leftarrow k_4^L||k_7^R, wk_3 \leftarrow k_7^L||k_4^R$;
for $i \leftarrow 0$ to 61 do
    if $(i+2) mod 8 = 0$ then
        $(k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7) \leftarrow (k_2, k_1, k_6, k_7, k_0, k_3, k_4, k_5)$;
    $rk_i \leftarrow k_{(i+2)mod8} \oplus con_i^{128}$,;
```

In both algorithms, k_i^L and k_i^R are the left and right half byte of k_i. In Algorithm 5, $(con_{2i}^{80}||con_{2i+1}^{80})$ is calcualted as $(con_{2i}^{80}||con_{2i+1}^{80}) \leftarrow (c_{i+1}||c_0||c_{i+1}||00_2 \ ||c_{i+1}||c_0||c_{i+1}) \oplus 0f1e2d3c_{16}$, where c_i is a 5-bit representation of i. In algorithm 2, we have $(con_{2i}^{128}||con_{2i+1}^{128}) \leftarrow (c_{i+1}||c_0||c_{i+1}||00_2||c_{i+1}||c_0||c_{i+1}) \oplus 6547a98b_{16}$.

We measure the memory complexity of our attacks as 64-bit Piccolo blocks and the time complexity in terms of the equivalent number of reduced-round Piccolo encryptions.

3.3.2 A MitM Attack on 14-Round Piccolo-80

In Piccolo, by noting that when the δ-set (see Definition 5) is chosen at the second input branch of the FN and the corresponding ordered sequence is evaluated at its first output branch, a distinguisher that minimizes the number of parameters can be constructed. However, such distinguisher does not lead to the best attack on Piccolo-80 since it can be extended upwards in the plaintext direction by two rounds only. If a third round is appended, the full codebook is needed due to the diffusion transformation utilized in Piccolo. Hence, to increase the number of rounds appended on top of the distinguisher, the δ-set is chosen at the first (instead of the second) input branch of the FN which, unfortunately, increases the number of parameters by two additional parameters. Then, in order to reduce the number of parameters, we exploit the properties of the diffusion operation M. In particular, we choose the δ-set to be after the first S-box layer of the first F-function such that after the diffusion transformation, only two nibbles are active, as shown in Figure 3.7. By enumerating all the possible values of three active input nibbles of the linear diffusion, it was found that such δ-set that has three active nibbles at the input of the linear transformation, and two active nibbles at its output contains 15 differences. Such δ-set enables us to build a 5-round distinguisher and overcome the problem of the two additional parameters when the δ-set is chosen at the first branch of the FN, as depicted in Figure 3.7, and captured by the following proposition:

Proposition 4 *Consider the encryption of a δ-set $\{Y^0 = P'^0[0:3]||P^0[4:15], Y^1 = P'^1[0:3]||P^1[4:15], \cdots, Y^{15} = P'^{15}[0:3]||P^{15}[4:15]\}$ through 5 rounds of Piccolo. The ordered sequence $[X_5^0[14:15] \oplus X_5^1[14:15], X_5^0[14:15] \oplus X_5^2[14:15], \cdots, X_5^0\ [14:15] \oplus X_5^{15}[14:15]$ is fully determined by the following 5 16-bit parameters, $X_0^0[0:3]$, $X_1^0[8:11]$, $X_2^0[0:3]$, $X_2^0[8:11]$ and $X_3^0[0:3]$.*

The above proposition means that we have $2^{5\times16} = 2^{80}$ ordered sequences out of the $2^{15\times8} = 2^{120}$ theoretically possible ones.

Proof. The knowledge of the δ*-set* $= \{Y^0, Y^1, \cdots, Y^{15}\}$ allows us to determine $[Y^0 \oplus Y^1, Y^0 \oplus Y^2, \cdots, Y^0 \oplus Y^{15}]$. In the sequel, we show that the ordered sequence at $X_5[14:15]$ can be determined uniquely by the knowledge of the 5 16-bit parameters mentioned in Proposition 4. As the δ-set is chosen at the input of the linear transformation M, it has to be propagated forward through M and backward through the first S-box layer to be able to determine the difference $\Delta X_1[6:7, 10:11, 13]$. To do this, we need to know three nibbles after the first S-box layer and two nibbles before the second S-box layer of the first F-function in the first round. However, the knowledge of only 4 nibbles $X_0^0[0:3]$ suffices to bypass the F-function and to compute $\Delta X_1[6:7, 10:11, 13]$. It is to be noted that only two nibbles are active after the F-function due to the restriction we place on the choice of our δ-set. Then, we bypass the second round by the knowledge of $X_1^0[8:11]$ which allows us to compute $\Delta X_2[2:3, 8:11, 14:15]$. By repeating the previous steps and propagating the differences further, $\Delta X_5[14:15]$ is computed. It is worth noting that there are nibbles which should have difference but appear in Figure 3.7 as if they do not have any difference, because their knowledge do not impact the computation of the ordered sequence at $X_5[14:15]$. For instance, after the third (resp. fourth) round, the difference at $X_3[8:11]$ (resp. $X_4[0:1]$) should be non-zero because the difference at $X_2[2:3, 8:11]$ (resp. $X_3[0:3]$) is non-zero.

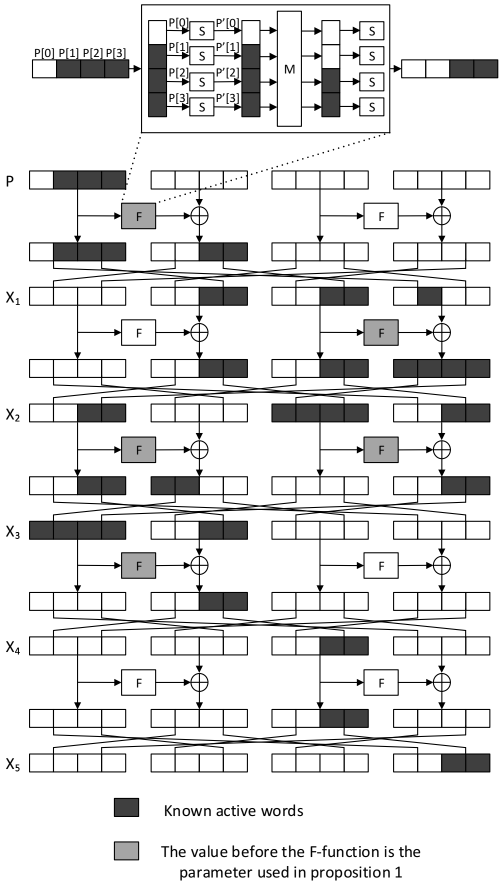


Figure 3.7: 5-round distinguisher to attack 14-round Piccolo-80

In what follows we show how to utilize the above described distinguisher to attack 14-round Piccolo-80 starting from the 5th round (round 4) till the 18th round (round 17) without the pre-whitening or the post-whitening keys. The attack relies on the previous proposition and exploits the linearity of the key schedule to build a 5-round distinguisher and then append 4 rounds above it and 5 rounds below it, as seen in Figure 3.8. The attack has two phases as follows:

Offline Phase. As demonstrated in Proposition 4, we determine all the 2^{80} ordered sequences and store them in a hash table H.

Online Phase. The online phase, as seen in Figure 3.8, proceeds as follows:

1. A plaintext P^0 is chosen as a reference to all the differences in the δ-set.

2. The δ-set P^0, P^1, $\cdots$, P^{15} is determined by guessing the state variables $X_6^0[8:11]$, $X_6^0[6:7, 12:13]$, $X_6^0[4:5, 14:15]$, and $X_8^0[1:3]$ to decrypt the δ-set differences.

3. The corresponding ciphertexts C^0, C^1, $\cdots$, C^{15} are requested.

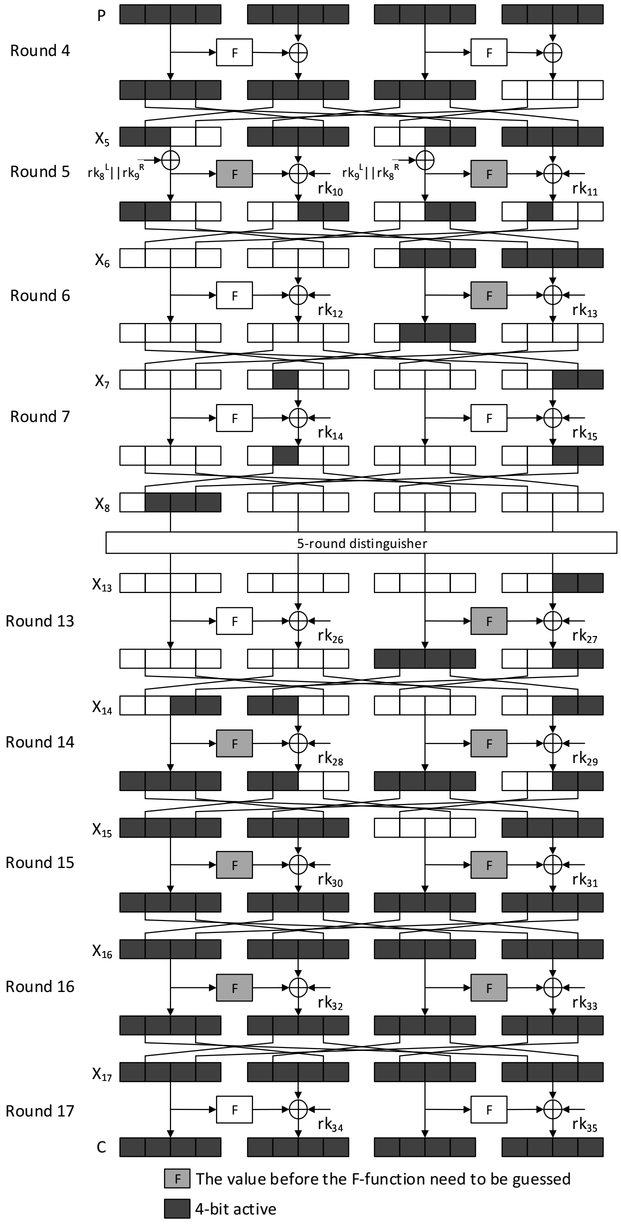


Figure 3.8: 14-round attack on Piccolo-80

4. The ordered sequence differences $[X_{13}^0[14:15] \oplus X_{13}^1[14:15], X_{13}^0[14:15] \oplus X_{13}^2[14:15], \cdots, X_{13}^0[14:15] \oplus X_{13}^{15}[14:15]]$ are determined by guessing the state variables $X_{13}^0[8:11]$, $X_{14}^0[0:3]$, $X_{14}^0[8:11]$, $X_{15}^0[0:3]$, $X_{15}^0[8:11]$, $X_{16}^0[0:3]$, $X_{16}^0[8:11]$ that are required to decrypt the ciphertext differences $[C^0 \oplus C^1, C^0 \oplus C^2, \cdots, C^0 \oplus C^{15}]$.

5. The guessed state variables are filtered by checking if the computed ordered sequence exists in H or not.

The evaluation of the δ-set and the corresponding ordered sequence as demonstrated in steps 2 and 4 require the guessing of 43 internal state nibbles. Guessing these 43 internal state nibbles makes the attack complexity exceeds the exhaustive search. Therefore, we analyze the key schedule searching for relations between the round keys to reduce the number of guessed parameters. As a result, we find that starting the attack from the 5^{th} round, i.e., round 4 is the best choice to reduce the number of the guessed parameters. Indeed, by only guessing k_0, k_1, k_2, k_3 and with the knowledge of P^0, we are able to compute $X_6^0[8:11]$, $X_6^0[6:7, 12:13]$, $X_6^0[4:5, 14:15]$, and $X_8^0[1:3]$. The knowledge of $[C^0, C^1, \cdots, C^{15}]$, $[C^0 \oplus C^1, C^0 \oplus C^2, \cdots, C^0 \oplus C^{15}]$ and the same keys guessed above enables us to evaluate the state variables $X_{13}^0[8:11]$, $X_{14}^0[0:3]$, $X_{14}^0[8:11]$, $X_{15}^0[0:3]$, $X_{15}^0[8:11]$, $X_{16}^0[0:3]$, $X_{16}^0[8:11]$. Consequently, we have to guess 4 round keys (16 nibbles) instead of guessing 43 internal state nibbles. Moreover and in order to reduce the memory complexity of the attack even further, we choose to compute the ordered sequence at only 6-bit instead of 8-bit, where any arbitrary 6-bit from the 8-bit can be chosen. Therefore, the probability of a wrong key to be a key candidate is $2^{80-(15\times 6)} = 2^{-10}$. As we have 2^{64} keys to be guessed, we expect that only $2^{64-10} = 2^{54}$ keys to remain after step 5. Hence, to recover the master key we guess k_4 and test the 2^{54} key candidates along with k_4 with just two plaintext/ciphertext pairs.

Attack Complexity. The memory complexity is determined by the size of the hash table H created in the offline phase. This table contains 2^{80} ordered sequences, where each ordered sequence has 15 6-bit differences. Therefore, the memory complexity is $2^{80} \times 90/64 = 2^{80.49}$ 64-bit blocks. To reduce the memory complexity below 2^{80}, we use a simple tradeoff and store a fraction $1/\alpha$ of H and repeat the attack α times as now we have decreased the chance to hit one element in H. We choose $\alpha = 2^7$ to reduce the memory complexity while still having a non-marginal time complexity. Hence, the memory complexity of the attack is $2^{73.49}$. As depicted in Figure 3.8, we shift the round keys rk_8, rk_9 from the 5^{th} round to the 6^{th} round. This round keys shift enable us to append 4 rounds, and not just 3 rounds, on top of our 5-round distinguisher with the same data complexity and without requiring the full codebook. To illustrate how this is possible, we choose our plaintexts such that after the 5^{th} round the words $X_5[2:3, 8:9]$ take a fixed value while the remaining words of X_5

take all the possible values. Hence, the data required can be formed using one structure that contains 2^{48} states of X_5. In order to obtain its corresponding plaintexts, we simply decrypt this structure as no keys are involved in this round any more. Accordingly, the data complexity is upper bounded by 2^{48} chosen plaintexts. Repeating the attack 2^7 times does not increase the data complexity as we just choose a different reference plaintext P^0. The time complexity of the offline phase is determined by the time needed to build the hash table H that now contains 2^{73}, instead of 2^{80}, ordered sequences. Therefore, the time complexity of the offline phase is $2^{73} \times 16 \times 5/(2 \times 14) = 2^{74.51}$. The time complexity of the online phase consists of two parts: the time required to filter the key space which is estimated to be $2^7 \times 2^{64} \times 16 \times (6 + 9)/(2 \times 14) = 2^{74.1}$ and the time to recover the master key which is estimated to be $2 \times 2^{(64-10)} \times 2^{16} = 2^{71}$. Hence, the total time complexity of the attack is $2^{74.51} + 2^{74.1} + 2^{71} \approx 2^{75.39}$ 14-round Piccolo-80 encryptions.

3.3.3 A MitM Attack on 16-Round Piccolo-128

Reusing the ideas of the attack on Piccolo-80 does not lead to the best attack on Piccolo-128 because the key schedule of the latter is different. Therefore, we use the key dependent sieving technique in order to build a longer distinguisher with the least number of parameters. As depicted in Figure 3.9, we construct a 7-round distinguisher, that we employ to attack 16-round Piccolo-128 from the 2^{nd} round (round 1) to the 17^{th} round (round 16) with the post-whitening keys. The δ-set of our 7-round distinguisher is chosen to be active at $P[7]$ and our distinguisher is built using Proposition 5.

Proposition 5 *Consider the encryption of a δ-set $\{P^0, P^1, \cdots, P^{15}\}$ through 7 rounds of Piccolo. The ordered sequence $[X_7^0[13:15] \oplus X_7^1[13:15], X_7^0[13:15] \oplus X_7^2[13:15], \cdots, X_7^0[13:15] \oplus X_7^{15}[13:15]]$ is fully determined by the following 8 16-bit parameters $X_1^0[8:11]$, $X_2^0[0:3]$, $X_2^0[8:11]$, $X_3^0[0:3]$, $X_3^0[8:11]$, $X_4^0[0:3]$, $X_4^0[8:11]$ and $X_5^0[0:3]$.*

The previous 5-round distinguisher of our attack on 14-round Piccolo-80 is independent of the round keys while our 7-round distinguisher that we utilize to attack 16-round Piccolo-128 uses the round keys to reduce the number of parameters. Assuming that we know the internal state X_3^0, i.e., 4 parameters, and the round keys rk_6, rk_7, we can evaluate X_4^0. Therefore, the 6 F-functions from the third round to the fifth round of the distinguisher can be bypassed. To bypass the other two F-functions, we need to know rk_4^L, rk_5^R, rk_8^L, and rk_9^R only. If rk_4, rk_8 depend on the same k_i and rk_5, rk_9 rely on the same k_j then we can bypass the other two F-functions by guessing only k_i^L, and k_j^R. In such case, we can bypass the 8 F-functions of our 7-round distinguisher by guessing 7 parameters only. By placing our distinguisher to cover from the 5^{th} round (round 4) to the 11^{th} round (round 10), the number of parameters

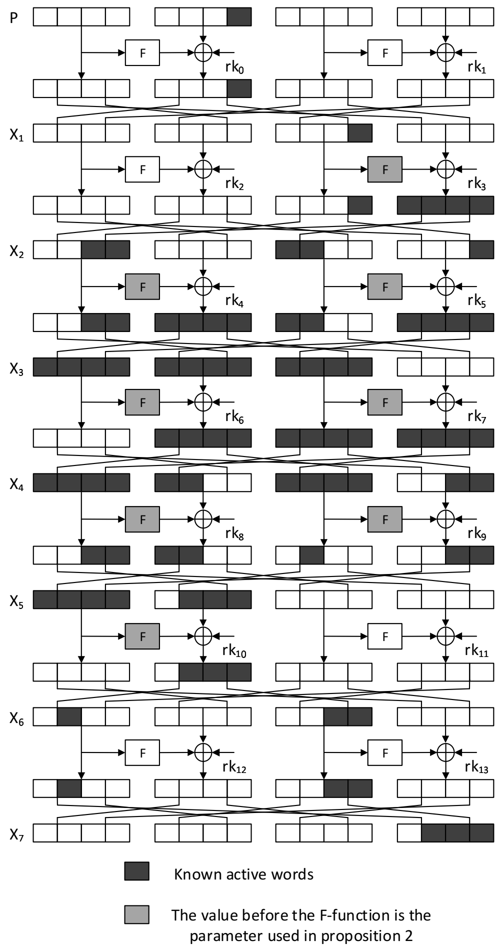


Figure 3.9: 7-round distinguisher to attack 16-round Piccolo-128

in Proposition 5 is reduced to 7 parameters only. In that case, k_i is k_4 and k_j is k_5 and the 7 16-bit parameters of our distinguisher are the state X_7^0, k_1, k_6, k_4^L, and k_5^R. Our 16-round attack is then built by appending 3 and 6 rounds at the top and the bottom of our 7-round distinguisher, respectively. As shown in Figure 3.10, the attack follows the same steps as the previous attack on Piccolo-80 while considering the new position of the δ-set at $X_4[7]$ and the different position of the corresponding ordered sequence at $X_{11}[13:15]$. In the online phase, the knowledge of P^0 and the guessing of k_4, k_5, k_6^L, and k_7^R enable us to partially decrypt $X_4[7]$ and determine the δ-set. From the other direction, by the knowledge of [C^0, C^1, $\cdots$, C^{15}], $[C^0 \oplus C^1, C^0 \oplus C^2, \cdots, C^0 \oplus C^{15}]$ and the guessing of k_0, k_1, k_2, k_3^R, k_5, k_6, and k_7, we can compute the ordered sequence at $X_{11}[13:15]$. Hence, in total we need to guess seven and half keys, i.e., k_0, k_1, k_2, k_3^R, k_4, k_5, k_6, and k_7, in order to mount our attack on 16-round Piccolo-128.

Attack Complexity. The memory complexity is estimated to be $2^{7\times16} \times (15 \times 12)/64 \approx 2^{113.49}$ 64-bit blocks and the data complexity is 2^{48} chosen plaintexts. The time complexity is

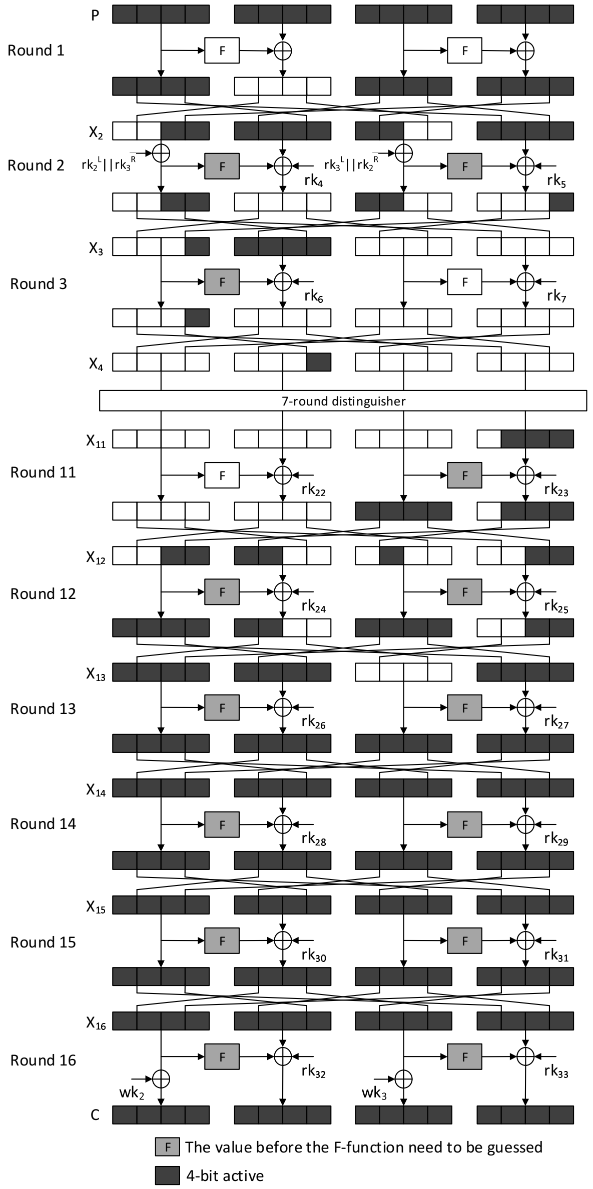


Figure 3.10: 16-round attack on Piccolo-128

$2^{112}\times 16\times 8/(2\times 16)+2^{120}\times 16\times (5+11)/(2\times 16)+2\times 2^{(120-68)}\times 2^{8} = 2^{114}+2^{123}+2^{61}\approx 2^{123}$ 16-round Piccolo-128 encryptions.

3.3.4 A MitM Attack on 17-Round Piccolo-128

To extend the attack on Piccolo-128 by one more round, we have to build another distinguisher, as illustrated in Figure 3.11 because using the previous 7-round distinguisher requires the guessing of the whole key space. Using this new 6-round distinguisher, which needs 8 parameters, we attack 17-round Piccolo-128 from the 5^{th} round (round 4) to the 21^{st} round (round 20) with the post-whitening keys. We append 4 and 7 rounds at the top and the bottom of our 6-round distinguisher, respectively. To launch the attack on 17-round

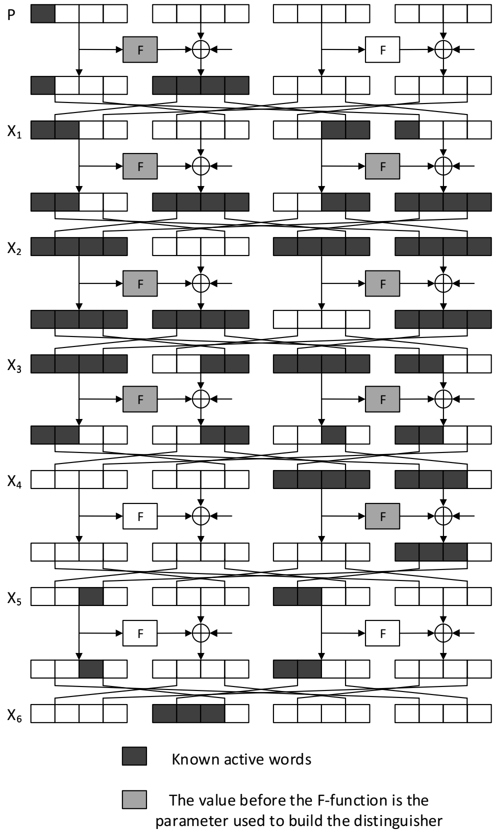

P
F
X₁
X₂
X₃
X₄
X₅
X₆
Known active words
The value before the F-function is the parameter used to build the distinguisher

Figure 3.11: 6-round distinguisher to attack 17-round Piccolo-128

Piccolo-128, we need to guess seven and half keys, as shown in Figure 3.12. These keys are $k_0^R, k_1, k_2, k_3, k_4, k_5, k_6, k_7$. The attack procedure follows the same steps of the previous attacks.

Attack Complexity. The memory complexity is estimated to be $2^{8\times16} \times (15 \times 12)/64 \approx 2^{129.49}$ 64-bit blocks. Since the memory complexity exceeds 2^{128}, we store a fraction $1/\alpha$ of the hash table H. $\alpha = 2^{3.5}$ is chosen so that the memory complexity does not exceed 2^{128} while having a non-marginal time complexity. Therefore, the memory complexity is $2^{125.99}$ 64-bit blocks. The data complexity is 2^{48} chosen plaintexts. Regarding the time complexity, since we store a fraction of the hash table, we have to repeat the online attack $2^{3.5}$ times. The time complexity of the offline phase is estimated to be $2^{128-3.5} \times 16 \times 8/(2 \times 17) \approx 2^{126.41}$. We use the partial computation technique in order to reduce the time complexity of the online phase. First, guessing the keys k_0^R, k_3^L, k_6, k_7 enables us to identify the δ-set and the time of this step is evaluated to be $2^{48} \times 16 \times 5/(2 \times 17) \approx 2^{49.23}$. By guessing k_4, k_5 we can partially decrypt through round 20 and this step is estimated to be $2^{80} \times 16 \times 2/(2 \times 17) \approx 2^{79.91}$. Then, guessing k_2 enables us to compute the output of the first F-function in round 19 and is estimated to be $2^{96} \times 16 \times 1/(2 \times 17) \approx 2^{94.91}$. Afterwards, guessing k_1 enables us to partially decrypt through round 19 and 18 as well as the first F-function of round 17 and needs

$2^{112} \times 16 \times 4/(2 \times 17) \approx 2^{112.91}$ encryptions. Finally, guessing k_3^R enables us to compute the ordered sequence and this step needs $2^{120} \times 16 \times 6/(2 \times 17) \approx 2^{121.5}$ encryptions. Accordingly, the time complexity of the online phase is $2^{49.23} + 2^{79.91} + 2^{94.91} + 2^{112.91} + 2^{121.5} \approx 2^{121.5}$ and it will be repeated $2^{3.5}$ times so, all in all, it is estimated to be 2^{125}. Recovering the master key using two plaintext/ciphertext pairs requires $2 \times 2^{120} \times 2^{128-180} \times 2^8 = 2^{77}$. The total time complexity of the attack is $2^{126.41} + 2^{125} + 2^{77} \approx 2^{126.87}$ encryptions.

3.4 Conclusion

In this chapter, we presented MitM attacks on Khudra. The time complexities of the attacks are given by $2^{66.11}$ and $2^{66.19}$ for the 13-round and 14-round reduced cipher, respectively. Both attacks have the same data and memory complexities of 2^{51} chosen plaintext and $2^{64.8}$ 64-bit blocks, respectively. Then, we presented MitM attacks on 14-round reduced Piccolo-80 and 16, 17-round reduced piccolo-128. All these attacks on Piccolo-80 and Piccolo-128 require the same data complexity of 2^{48} chosen plaintexts. The time complexities of the MitM attacks on 14-round Piccolo-80 and 16, 17-round Piccolo-128 are $2^{75.39}$, 2^{123}, and $2^{126.87}$, respectively. Their memory complexities are $2^{73.49}$, $2^{113.49}$, and $2^{125.99}$ for the 14-round Piccolo-80 and 16, 17-round Piccolo-128, respectively.

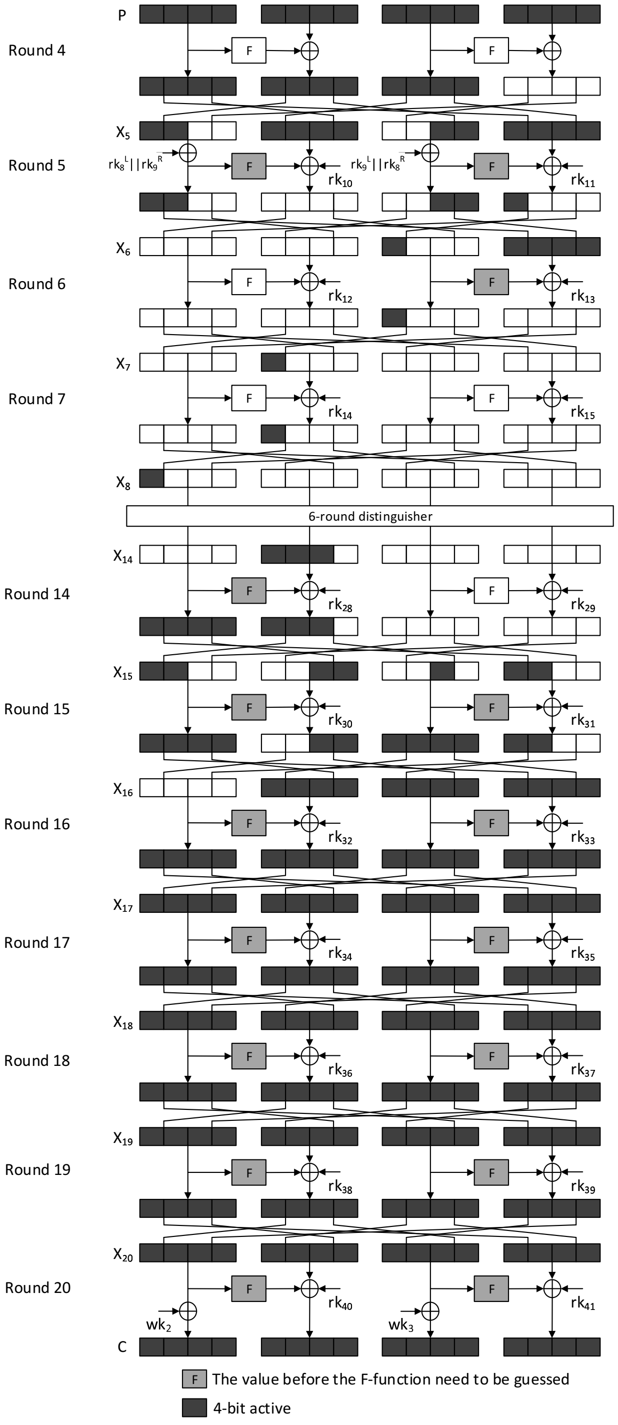


Figure 3.12: 17-round attack on Piccolo-128

Chapter 4

A MitM with Efficient Enumeration Attack on Kiasu-BC

Kiasu-BC is a recently proposed tweakable variant of the AES-128 block cipher. The designers of Kiasu-BC claim that no more than 7-round Meet-in-the-Middle (MitM) attack can be launched against it. In this chapter, we present a MitM attack, utilizing the efficient enumeration technique, on the 8-round reduced cipher. The attack has time complexity of 2^{116} encryptions, memory complexity of 2^{86} 128-bit blocks, and data complexity of 2^{116} plaintext-tweak combinations.

4.1 Introduction

Different from traditional block ciphers, a tweakable block cipher [96] has an additional input called a tweak. Varying this tweak allows the choice of different specific keyed instances of the cipher. Tweakable ciphers are designed such that changing the tweak can be done very efficiently compared to the key setup operation, which allows new interesting modes of operation and applications to become possible. Recently, the TWEAKY framework [72] was proposed by Jean *et al.* at AsiaCrypt 2014. In the extended version of [73], the authors proposed three specific instances of tweakable block ciphers including Kiasu-BC, which has also been used as a building block for the authenticated encryption scheme Kiasu [71] that was submitted to the CAESAR competition. Kiasu-BC is a tweakable version of AES-128 which has an additional 64-bit input tweak where, as shown in Figure 4.1, the tweak is added to the first two rows of every round key. Except for the tweak addition to the round keys, Kiasu-BC has the same specifications of AES-128. In [72, 73], the designers of Kiasu-BC claim that the 7-round Meet-in-the-Middle (MitM) attack against AES-128 is also applicable on Kiasu-BC without any extra rounds.

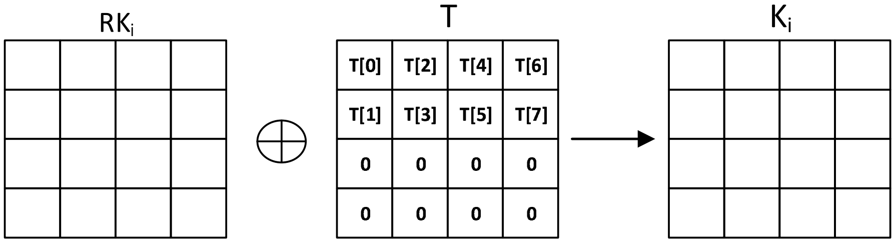


Figure 4.1: Tweak addition to the round key in Kaisu-BC, where RK_i is the AES-128 key of round $i-1$

In this chapter, we present a MitM attack on 8-round reduced version of Kiasu-BC, which contradicts the designers' claim. Our attack is based on the MitM with efficient enumeration and *b-δ-set* techniques [58, 67]. Using the additional degree of freedom available because of the ability to choose the tweak, we are able to extend the 4-round distinguisher of AES-128 to 5-round distinguisher against Kiasu-BC. Then, by appending one round above and two rounds below the distinguisher, we launch an attack against the 8-round reduced cipher.

The rest of the chapter is organized as follows. Section 4.2 provides the notations used throughout the chapter and a brief description of Kiasu-BC. In section 4.3, we present our attack on 8 rounds of Kiasue-BC. Finally, the chapter is concluded in section 4.4.

4.2 Specifications of Kiasu-BC

Kiasu-BC is a tweakable block cipher presented in [72, 73]. Then it was used as a building block for the authenticated encryption Kiasu [71] that was submitted to the CAESAR competition. Kiasu-BC accepts 128-bit plaintext blocks, 128-bit key and a 64-bit public tweak T.

As mentioned above, the specification of Kiasu-BC is essentially identical to AES-128, except that T is XORed to the first two rows of every round key. When $T = 0$, Kiasu-BC is equivalent to AES-128. In other words, apart from the tweak, it has the same specifications as AES-128. Similar to AES-128, each round of Kiasu-BC has four operations: SubBytes (SB), ShiftRows (SR), MixColumns (MC) and AddRoundTweakKey (ATK). In what follows, we give a brief description of these operations:

- SubBytes (SB): A nonlinear byte bijective mapping.
- ShiftRows (SR): Rotation of row i of the state, $0 \leq i \leq 3$, to the left by i bytes.
- MixColumns (MC): Each column of the state is multiplied by MDS matrix to obtain the corresponding column.

- AddRoundTweakKey (AK) : At round i, the state is XORed with the 128-bit round key RK_i and 128-bit τ, such that the first two rows contain the 64-bit tweak T and the other two rows contain zero values. Note that, $k_i = RK_i \oplus \tau$.

An initial AddRoundTweakKey operation is applied prior to the first round and the MixColumn operation in the last round is omitted. In our attack, we use the differential property of the S-box (see Proposition 1). For further details regarding the block cipher specification, the reader is referred to [72, 73].

The following notation is used throughout the remaining of this chapter:

- x_i: The 16-byte state before the SB operation at round i.
- y_i: The 16-byte state before the SR operation at round i.
- z_i: The 16-byte state before the MC operation at round i.
- w_i: The 16-byte state before the ATK operation at round i.
- k_i: The XOR value of the round key and the tweak at round $i - 1$.
- x_i^j: The state at round i whose position within a set or a sequence is given by j.
- $x_i[j]$: The j^{th} byte of the state x_i, where $j = 0, 1, \cdots, 15$.
- $x_i[j, l]$: The bytes j and l of x_i.
- Δx_i, $\Delta x_i[j]$: The difference at state x_i, and byte $x_i[j]$, respectively.

The memory complexity of our attack is given in 128-bit blocks and the time complexity is evaluated in reduced round Kiasu-BC encryptions.

4.3 A MitM Attack on 8-Round Kiasu-BC

In our MitM attack, we utilize a *b-δ-set* (see Definition 6) to reduce the memory complexity where we set $b = 5$. The distinguisher is employed in the middle 5 rounds from round 1 to 5 such that the *b-δ-set* is presented at $w_0[0]$, the ordered sequences are observed at $x_6[0]$. By utilizing a similar idea to the one presented in [62], we are able to extend the distinguisher from 4 rounds (on AES-128) to 5 rounds on Kiasu-BC by exploiting the freedom in the choice of the tweak. In our distinguisher, the difference in the tweak is chosen at byte 0 and is set equal to the *b-δ-set* such that the difference at x_1 after the AddRoundTwekKey operation is 0, as illustrated in Figure 4.2. Proposition 6, below, is the core of our attack.

Proposition 6 *If a message m belongs to a pair of states conforming to the distinguisher of Figure 4.2, then the ordered sequences of differences $\Delta x_6[0]$ obtained from the b-δ-set constructed from m in $w_0[0]$ are fully determined by the following one 5-bit and 10 bytes: $\Delta w_0[0], x_2[0], x_3[0,1,2,3], y_5[0,5,10,15]$, and $\Delta x_6[0]$.*

Proof. The proof is based on the efficient enumeration technique [58]. In the following we will show how the knowledge of one 5-bit and 10 bytes is enough to propagate the *b-δ-set* at w_0 and compute the ordered sequences at x_6. Let (m, m') be a right pair that conforms to the distinguisher in Figure 4.2. Since $\Delta w_0[0] = \Delta k_1[0]$, then $\Delta x_1[0] = 0$. Also, since $\Delta k_2[0] = \Delta k_1[0]$, then $\Delta x_2[0] = \Delta w_0[0]$. The knowledge of $x_2[0]$ allows us to determine $\Delta y_2[0]$. Then $\Delta y_2[0]$ can be propagated through the following linear operations: SR, MC and ATK to compute $\Delta x_3[0,1,2,3]$. With the knowledge of $x_3[0,1,2,3]$, we can bypass the non-linear operation SB and then linearly we can forward the knowledge through SR, MC, and ATK to get Δx_4. Similarly, in the backward direction, we propagate $\Delta x_6[0]$ linearly through ATK, MC^{-1} and SR^{-1}. Then, the knowledge of $y_5[0,5,10,15]$ allows us to bypass the SB^{-1} to get $\Delta x_5[0,5,10,15]$ that can be propagated backward linearly through ATK, MC^{-1}, and SR^{-1} to compute Δy_4. Using the differential property of the S-boxes, the state x_4 has one solution, on average.

Attack Procedure: In what follows, we show how we can launch an 8-round attack on Kiasu-BC using the above distinguisher. The attack is constructed by appending one round on the top of the distinguisher and two rounds beneath it. The attack has two phases, namely the precomputation phase and the online phase.

Precomputation Phase. In this phase, we iterate over the $2^{10\times 8+5} = 2^{85}$ possible values for the one 5-bit and 10 bytes in Proposition 1, and for each of them we deduce the internal state variable x_4. Then, we build the table and in each entry we store the ordered sequences computed at $x_6[0]$. Therefore, we have 2^{85} ordered sequences out of the $2^{32\times 8=256}$ theoretically possible ones.

Online Phase. This phase is decomposed of two steps. The first step is the *data collection* in which we collect many pair of messages to guarantee that one of them confirm to the truncated differential characteristic in Figure 4.2. The second step is the *key recovery* in which the collected data pairs are used to create the *b-δ-set* to compute the ordered sequences in the online phase and compare it with precomutation table to identify the key candidates.

Data Collection. This type of the MitM attacks operates in the chosen plaintext model.

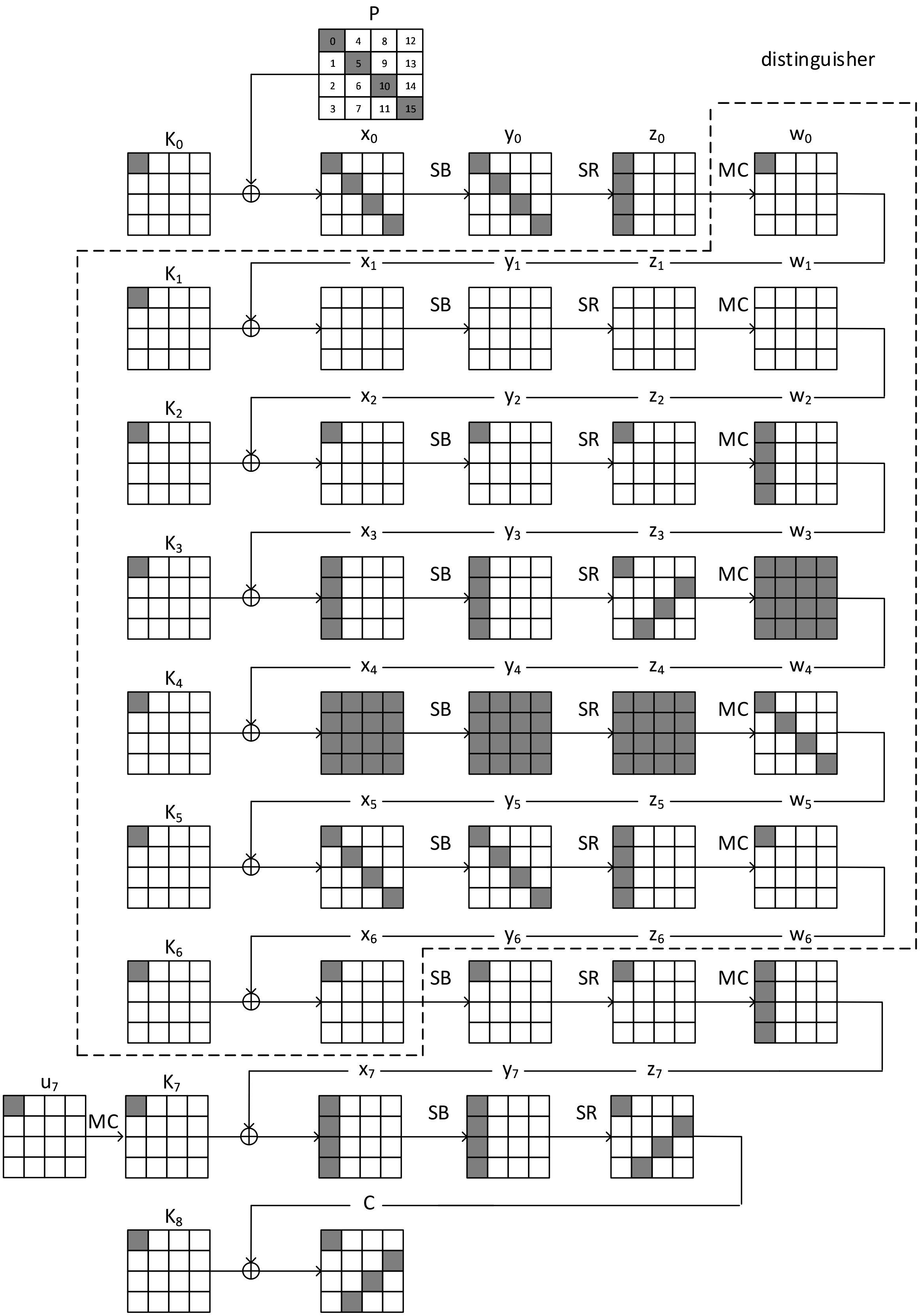


Figure 4.2: A MitM attack on 8-round Kiasu-BC

To reduce the number of required chosen plaintexts and get enough pairs to launch the attack, we use the structure technique. Here, our structure takes all the possible values in bytes $0, 5, 10, 15$, while all the remaining bytes are fixed. In addition, we choose the tweak such that the left most byte takes only 2^5 possible values. Therefore, one structure generates $2^{4\times8+5} \times (2^{4\times8+5} - 1)/2 \approx 2^{73}$ possible pairs with 2^{37} plaintext-tweak combinations. We can calculate the probability of the whole truncated differential characteristic from the following probabilities: transition $4 \to 1$ ($z_0 \to w_0$) over MC of probability 2^{-24}, AddRoundTweakKey cancellation of $w_0[0], k_1[0]$ of probability 2^{-8}, transition $4 \to 1$ $w_6 \to z_6$ over MC^{-1} of probability 2^{-24} and to have 12 zero differences in the ciphertext which happens with probability $2^{-12\times8=-96}$. Therefore, the total probability of the truncated differential characteristic is $2^{-24-8-24-96=-152}$. Hence, to find one pair of messages that confirm to the truncated differential characteristic we need to collect 2^{152} message pairs. Since each structure contains 2^{73} message pairs, we need 2^{79} structures. Thus, the data complexity of the attack is $2^{79+37=116}$ plaintext-tweak combinations. Therefore, we query the encryption oracle with 2^{116} plaintext-tweak combinations.

Key Recovery. In this step, we first try to identify the number of key suggestions of 9 bytes, $k_0[0, 5, 10, 15], u_7[0], k_8[0, 7, 10, 13]$ that correspond to each pair of messages. This can be achieved as follows: to deduce the values of the 4 bytes of $k_0[0, 5, 10, 15]$, we guess 2^5 possible values of $\Delta w_0[0]$ and propagate the values linearly through MC^{-1}, SR^{-1} to get $\Delta y_0[0, 5, 10, 15]$. The pairs of messages we have are grouped into 2^5 different groups based on the tweak difference at $\Delta k_0[0]$, which is equal to $\Delta w_0[0]$. Then, the knowledge of the plaintexts with the tweak $\Delta k_0[0]$ allows to compute $\Delta x_0[0, 5, 10, 15]$. Using the differential property of the S-box, we evaluate $x_0[0, 5, 10, 15]$. The knowledge of the plaintext with $x_0[0, 5, 10, 15]$ allows us to deduce the values of the 4 bytes $k_0[0, 5, 10, 15]$. To deduce the values of the 4 bytes $k_8[0, 7, 10, 13]$, we guess the byte $\Delta z_6[0]$ and propagate it linearly through MC and ATK to compute $\Delta x_7[0, 1, 2, 3]$. The knowledge of ciphertext allows us to compute $\Delta y_7[0, 1, 2, 3]$. Then, we can deduce the value of the four bytes $y_7[0, 1, 2, 3]$ using the differential property of the S-box. The knowledge of the ciphertext and $y_7[0, 1, 2, 3]$ allows the computation of $k_8[0, 7, 10, 13]$. The value of $u_7[0]$ can be deduced by guessing $\Delta x_6[0]$. To summarize this part, we have $2^{5+2\times8} = 2^{21}$ key candidates for the 9 key bytes $k_0[0, 5, 10, 15], u_7[0], k_8[0, 7, 10, 13]$.

From the 2^{152} message pairs required to satisfy the truncated differential characteristic, we only use 2^{56} message pairs to identify the *b-δ-set* and build the ordered sequences because we have 12 bytes filter in the ciphertext side. For every pair of the remaining message pairs and for every key value of the 9 key bytes we identify the *b-δ-set* and compute the ordered sequences at $x_6[0]$. If no match is found in the precomputation table, the key value is discarded.

Otherwise, it is considered as a key candidate. The probability of having a wrong key match in the table is $2^{85} \times 2^{-256} = 2^{-171}$. Therefore, after this step, we will have $2^{56+21-171} = 2^{-94}$ key candidates for the key bytes $k_0[0, 5, 10, 15], k_8[0, 7, 10, 13], u_7[0]$, i.e., we will end up with the correct key only.

Attack Complexity. The memory complexity of the attack is dominated by the precomputation phase which requires the storage of 2^{85} ordered sequences where each ordered sequence is 256 bits . Therefore, the memory complexity of the attack is $2^{85} \times 256/128 = 2^{86}$ 128-bit blocks. The data complexity of the attack is determined from the data collection step. Therefore, the data complexity is 2^{116} plaintext-tweak combinations. The time complexity of the offline phase, which is the time required to build the precomutation table, is $2^{85} \times 2^5 \times 2^{-4} \approx 2^{86}$ encryptions. The time complexity of the online phase is $2^{56} \times 2^{21} \times 2^5 \times 2^{-5} \approx 2^{77}$. Therefore, the time complexity of the attack is dominated by the data collection phase which is 2^{116} encryptions.

4.4 Conclusion

By utilizing the freedom in the tweak, we presented a MitM attack against 8-round reduced Kiasu-BC, which violates the designers' claim that no more than 7 rounds can be reached using MitM attacks.

Chapter 5

Impossible Differential Cryptanalysis of SKINNY

SKINNY is a new lightweight tweakable block cipher family proposed by Beierle *et al.* at CRYPTO 2016. SKINNY has 6 main variants where SKINNY-n-t is a block cipher that operates on n-bit blocks using t-bit tweakey (key and tweak) where $n = 64$ or 128 and $t = n$, $2n$, or $3n$. In this chapter, we present impossible differential attacks against reduced-round versions of all the 6 members of the SKINNY family in the single-tweakey model. More precisely, using an 11-round impossible differential distinguisher, we present impossible differential attacks against 18-round SKINNY-n-n, 20-round SKINNY-n-$2n$ and 22-round SKINNY-n-$3n$ ($n = 64$ or 128). The time, data and memory complexities of our attacks are presented in Table 5.1.

Table 5.1: The time, data and memory complexities of our attacks

Block cipher version	# of rounds	Time	Data	Memory
SKINNY-64-64	18	$2^{57.1}$	$2^{47.52}$	$2^{58.52}$
SKINNY-128-128	18	$2^{116.94}$	$2^{92.42}$	$2^{115.42}$
SKINNY-64-128	20	$2^{121.08}$	$2^{47.69}$	$2^{74.69}$
SKINNY-128-256	20	$2^{245.72}$	$2^{92.1}$	$2^{147.1}$
SKINNY-64-192	22	$2^{183.97}$	$2^{47.84}$	$2^{74.84}$
SKINNY-128-384	22	$2^{373.48}$	$2^{92.22}$	$2^{147.22}$

5.1 Introduction

SKINNY [24] is a Substitution Permutation Network (SPN) family of tweakable lightweight block ciphers proposed at CRYPTO 2016 by Beierle *et al.* It supports two block lengths

$n = 64$ or 128 and for each of them, the tweakey t can be either $n, 2n$ or $3n$. This family of ciphers inherits the recent design trend of having an SPN cipher with suboptimal internal components. More precisely, SKINNY uses a light tweakey schedule along with a round function that consists of a compact S-box and a sparse diffusion layer. However, these suboptimal components are arranged such that tight security bounds are guaranteed. Indeed, using Mixed Integer Linear Programming (MILP), the designers of SKINNY provide high security bounds against differential/linear attacks for all the SKINNY versions in both the single-tweakey and related-tweakey models. Furthermore, SKINNY has a good performance for round-based ASIC implementation as it requires a very small area using serial ASIC. Moreover, the designers of SKINNY show that its ASIC threshold implementation is very favorable to AES-128 threshold implementation [31]. Providing compact implementation and a high level of security with the existence of the tweakey was feasible by generalizing the Superposition TWEAKEY (STK) construction [72]. Lastly, being a tweakable block cipher allows SKINNY to be employed into a higher level of operating modes such as SCT [112].

The designers of SKINNY presented 16-round attacks against SKINNY-n-n ($n = 64$ or 128) in the single-tweakey model utilizing 11-round impossible differential distinguisher. To provoke public cryptanalysis of SKINNY, they have announced a competition [23] against two particular variants of SKINNY, namely, SKINNY-64-128 and SKINNY-128-128, in which they indicated that the best known attack against SKINNY-64-128, in the single-tweakey model, is 18 rounds. As a result, a handful of third-party analysis have been published [97, 114, 15]. However, these attacks are in the arguably weaker attack model, the related-tweakey model, in which the attacker is assumed to have the ability to query the encryption oracle with keys that have specific relations.

In this chapter, we present impossible differential attacks against reduced-round versions of all the 6 variants of SKINNY, namely, SKINNY-n-n, SKINNY-n-$2n$ and SKINNY-n-$3n$ ($n = 64$ or 128). All these attacks utilize the same 11-round impossible differential distinguisher. Then, we exploit the fact that the tweakey additions are only performed on the first two rows of the state, along with the MixColumns operation properties and the tweakey schedule relations, to extend this distinguisher by 7, 9, 11 rounds to launch key-recovery attacks in the single-tweakey model against 18, 20, 22 rounds of SKINNY-n-n, SKINNY-n-$2n$ and SKINNY-n-$3n$ ($n = 64$ or 128), respectively. Specifically, we extend the designers' 11-round impossible differential distinguisher by 3, 3 and 3 rounds above it and 4, 6 and 8 rounds below it to launch 18, 20 and 22 rounds attacks against SKINNY-n-n, SKINNY-n-$2n$ and SKINNY-n-$3n$ ($n = 64$ or 128), respectively.

The rest of the chapter is organized as follows. Section 5.2 provides the notations used throughout the chapter and a brief description of SKINNY . In section 5.3, we present the impossible differential distinguisher used in our attacks. The details of our attacks are presented in sections 5.4, 5.5 and 5.6, respectively. Finally, the chapter is concluded in section 5.7.

5.2 Specifications of SKINNY

The following notations are used throughout the rest of the chapter:

- TK_i: The round tweakey used in round i.
- ETK_i: The equivalent round tweakey used in round i.
- x_i: The input to the SubCells (SC) operation at round i.
- y_i: The input to the AddRoundConstantTweakey (AK) operation at round i.
- $y_i^{'}$: The input to the AddRoundConstantEquivlantTweakey (AEK) operation at round i.
- z_i: The input to the ShiftRows (SR) operation at round i.
- w_i: The input to the MixColumns (MC) operation at round i.
- $x_i[j]$: The j^{th} cell of x_i, where $0 \leq j < 16$.
- $x_i[j \cdots l]$: The cells from j to l of x_i, where $j < l$.
- $x_i[j, l]$: The cells j and l of x_i.
- $x_i[j][k]$: The k^{th} bit of the j^{th} cell of x_i.
- $x_i[j]\{k, l, m\}$: The XOR of bits k, l, m of cell j of x_i.
- $x_i[col : j]$: The four cells in column j, e.g., $x_i[col : 0] = x_i[0, 4, 8, 12]$.
- $x_i[SR^{-1}[col : j]]$: The four cells in column j after the SR^{-1} operation is applied, e.g., $x_i[SR^{-1}[col : 0]] = x_i[0, 7, 10, 13]$.
- $x_i[col : j][k, l]$: The j^{th} and l^{th} cells of column j of x_i, e.g., $x_i[col : 0][0, 1] = x_i[0, 4]$.
- $\Delta x_i, \Delta x_i[j]$: The difference at state x_i and cell $x_i[j]$, respectively.

Table 5.2: Number of rounds for SKINNY-n-t, with n-bit state and t-bit tweakey state

Block size n	Tweakey size t		
	n	$2n$	$3n$
64	32	36	40
128	40	48	56

The SKINNY family supports two block lengths of $n = 64$ and 128 bits. In both versions, the internal state *IS* is represented as a 4×4 array of cells such that one cell represents a nibble (when the block length $n = 64$) and a byte (when the block length $n = 128$). While classical block ciphers have two inputs, namely the plaintext and the key, and output the ciphertext, SKINNY is a tweakable block cipher [96, 72] that uses an input called the tweakey instead of the key. Then, the user has the freedom to choose which part of the tweakey to be assigned to the key and which part to be assigned to the tweak. This family of block ciphers with block length n deploys three main tweakeys of lengths $t = n$ bits, $t = 2n$ bits and $t = 3n$ bits. Similar to the state, the tweakey state can be represented as z 4×4 arrays of cells, i.e., we have arrays *TK1* (in case $z = 1$), *TK1* and *TK2* (in case $z = 2$), *TK1*, *TK2*, and *TK3* (in case $z = 3$).

The encryption operation proceeds as follows. First, the plaintext $m = m_0 \| m_1 \| \cdots \| m_{14} \| m_{15}$ (where $|m_i| = n/16 = s$-bit) is loaded into the internal state *IS* row-wise as depicted in Figure 5.1. Then, the tweakey input $tk = tk_0 \| tk_1 \| \cdots \| tk_{16z-1}$ (where $|tk_i|$ is s-bit as in the internal state) is loaded row-wise such that $TK1[i] = tk_i$ for $0 \leq i \leq 15$ (in case $z = 1$), $TK1[i] = tk_i$, $TK2[i] = tk_{16+i}$ for $0 \leq i \leq 15$ (in case $z = 2$) or $TK1[i] = tk_i$, $TK2[i] = tk_{16+i}$, $TK3[i] = tk_{32+i}$ for $0 \leq i \leq 15$ (in case $z = 3$). Finally, the internal state is updated by applying the round function r times, where the number of rounds r depends on the block length and the tweakey size as shown in Table 5.2.

As shown in Figure 5.1, in each round, SKINNY applies five different operations, namely, SubCells, AddConstants, AddRoundTweakey, ShiftRows and MixColumns. The cipher does not apply whitening tweakeys. Consequently, parts of the first and last rounds do not add any security. In what follows, we describe the five different operations that are employed in each round:

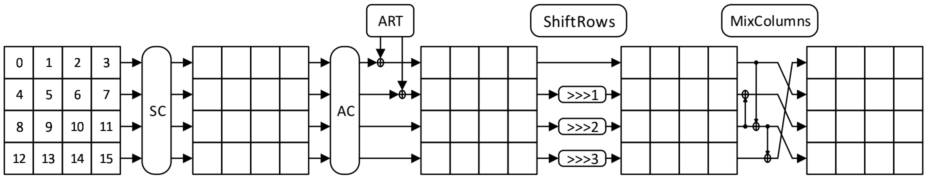


Figure 5.1: The SKINNY round function

Table 5.3: The SKINNY LFSR used in the tweakey schedule, where s denotes the cell size in bits

TK	s	LFSR
*TK*2	4	$(x_3 \parallel x_2 \parallel x_1 \parallel x_0) \rightarrow (x_2 \parallel x_1 \parallel x_0 \parallel x_3 \oplus x_2)$
	8	$(x_7 \parallel x_6 \parallel x_5 \parallel x_4 \parallel x_3 \parallel x_2 \parallel x_1 \parallel x_0) \rightarrow (x_6 \parallel x_5 \parallel x_4 \parallel x_3 \parallel x_2 \parallel x_1 \parallel x_0 \parallel x_7 \oplus x_5)$
*TK*3	4	$(x_3 \parallel x_2 \parallel x_1 \parallel x_0) \rightarrow (x_0 \oplus x_3 \parallel x_3 \parallel x_2 \parallel x_1)$
	8	$(x_7 \parallel x_6 \parallel x_5 \parallel x_4 \parallel x_3 \parallel x_2 \parallel x_1 \parallel x_0) \rightarrow (x_0 \oplus x_6 \parallel x_7 \parallel x_6 \parallel x_5 \parallel x_4 \parallel x_3 \parallel x_2 \parallel x_1)$

- SubCells (*SC*): A nonlinear bijective mapping applied on every cell of the internal state, where 4-bit (in case $n = 64$) or 8-bit (in case $n = 128$) S-boxes are applied.

- AddConstants (*AC*): A 4×4 round constant is XORed to the state. These round constants are generated using a 6-bit affine LFSR. The details of generating the round constants can be found in [24].

- AddRoundTweakey (*ART*): The first and second rows of all the tweakey arrays are XORed to the state. More precisely, for $0 \leq i \leq 7$, we have:

  - $IS[i] = IS[i] \oplus TK1[i]$, when $z = 1$,
  - $IS[i] = IS[i] \oplus TK1[i] \oplus TK2[i]$, when $z = 2$,
  - $IS[i] = IS[i] \oplus TK1[i] \oplus TK2[i] \oplus TK3[i]$, when $z = 3$.

- ShiftRows (*SR*): The rows of the state are rotated as in AES but to the right, i.e., the following permutation $P = [0, 1, 2, 3, 7, 4, 5, 6, 10, 11, 8, 9, 13, 14\ , 15, 12]$ is applied.

- MixColumns (*MC*): Each column in the state is multiplied by a binary matrix M, where M and its inverse M^{-1} are given as follows:

$$M = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}, \; M^{-1} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}.$$

Tweakey Schedule. As depicted in Figure 5.2, the tweakey arrays are updated through tweakey schedule as follows. First all the tweakey arrays, i.e., *TK1* (when $z = 1$), *TK1*, *TK2* (when $z = 2$), or *TK1*, *TK2*, *TK3* (when $z = 3$) are permuted using a permutation P_T such that $P_T = [9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6\ , 7]$. Finally, each cell in the first and second rows of *TK2*, *TK3* (when $z = 2$ or $z = 3$) is updated using the LFSR operations shown in Table 5.3, where x_0 is the LSB of the cell.

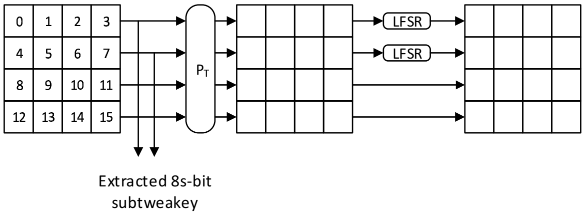


Figure 5.2: The tweakey schedule

In our attack, we use AddKey (AK) operation which compromises the AC and ART operations. Moreover, we swap the linear operations AK, $MC \circ SR$, and hence we use the equivalent subtweakey ETK instead of the subtweakey TK such that $ETK_{r+1} = MC \circ SR(TK_r)$.

5.3 An Impossible Differential Distinguisher of SKINNY

The designers of SKINNY exhaustively searched for the longest truncated impossible differential that has one active cell in both $\Delta\delta$ and $\Delta\beta$. They found 16 such truncated impossible differentials where each one covers 11 rounds. They exploited one of these 16 impossible differentials, illustrated in Figure 5.3, to attack 16-round SKINNY-n-n ($n = 64$ or 128). This distinguisher, which we reuse in our attacks, states that a pair of messages that has only one active cell at $x_3[12]$ cannot have only one active cell at $x_{14}[8]$. The reason is that the active cell $\Delta x_3[12]$ results in 4 active cells and 12 unknown cells after 6 rounds, i.e., at state x_9. From the other side, the active cell $\Delta x_{14}[8]$ results in 4 inactive cells, 5 unknown cells and 7 active cells at state Y_9 contradicting with the forward differential at $\Delta y_9[15]$.

Our attacks depend on the differential property of the S-box (see Proposition 1).

All our attacks use the same 11-round distinguisher, have 3 analysis rounds on its top. They, however, differ in the analysis rounds appended below it. In what follows, we describe our attack against SKINNY-64-128 in details and then mention only the main differences for the other attacks.

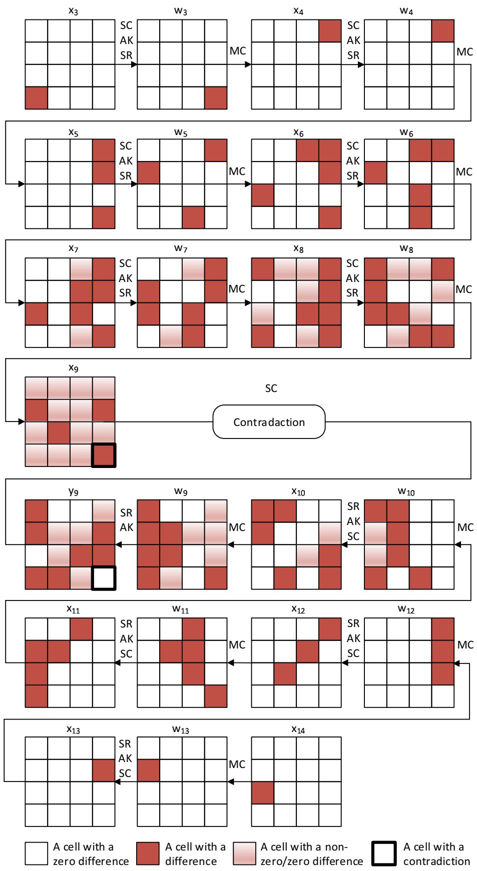


Figure 5.3: Impossible differential distinguisher of SKINNY

5.4 Impossible Differential Key-recovery Attack on 20-round SKINNY-n-2n

5.4.1 Impossible Differential Key-recovery Attack on SKINNY-64-128

In this section, we present the first published attack on 20-round SKINNY-64-128 in the single-tweakey model. We use the notion of data structures to generate enough pairs of messages to launch the attack. In the first three rounds, we use the equivalent tweakey ETK instead of the tweakey TK. Therefore, the first round has no tweakey, and hence we can build our structures at y_1'. Then, we propagate it backward linearly through MC^{-1}, SR^{-1}, and SC^{-1} to obtain the corresponding plaintexts. Our utilized structure takes all the possible values in 7 nibbles $y_1'[3, 4, 5, 6, 9, 11, 14]$ while the remaining nibbles take a fixed value. Thus, one structure generates $2^{4\times7} \times (2^{4\times7} - 1)/2 \approx 2^{55}$ possible pairs. Hence, we have 2^{55} possible pairs of messages satisfying the plaintext differences. In addition, we utilize the following pre-computation tables in order to efficiently extract/filter the (equivalent) tweakey nibbles corresponding to the active state nibbles involved in the analysis rounds, where the table

$H_l\{(E)TK_i[\mathbb{S}]\}$ (also referred to as H_l) is used to extract/filter the (equivalent) tweakey used in round i at cells belonging to the set $\mathbb{S}$ and H^* is computed once and used to extract all the tweakey nibbles of the last analysis round and those corresponding to column 1 in round 18.

$\boldsymbol{H_1\{TK_{18}[2,6]\}}$: For all the 2^{24} possible values of $\Delta z_{17}[SR^{-1}[col:2][0,1]]$, $z_{17}[SR^{-1}[col:2]]$, compute $\Delta y_{18}[col:2], y_{18}[col:2]$. Then, store $\Delta z_{17}[SR^{-1}[col:2][0,1]]$, $z_{17}[SR^{-1}[col:2]], y_{18}[col:2][0,1]$ in H_1 indexed by $\Delta y_{18}[col:2], y_{18}[col:2][2,3]$. H_1 has 2^{24} rows and on average about $2^{24}/2^{24}=1$ value in each row.

$\boldsymbol{H_2\{TK_{18}[0,4]\}}$: For all the 2^{28} possible values of $\Delta z_{17}[SR^{-1}[col:0][0,2,3]]$, $z_{17}[SR^{-1}[col:0]]$, compute $\Delta y_{18}[col:0], y_{18}[col:0]$. Then, store $\Delta z_{17}[SR^{-1}[col:0][0,2,3]]$, $z_{17}[SR^{-1}[col:0]], y_{18}[col:0][0,1]$ in H_2 indexed by $\Delta y_{18}[col:0]$, $y_{18}[col:0][2,3]$. H_2 has 2^{24} rows and on average about $2^{28}/2^{24}=2^4$ values in each row.

$\boldsymbol{H_3\{TK_{18}[3,7]\}}$: For all the 2^{28} possible values of $\Delta z_{17}[SR^{-1}[col:3][0,1,3]]$, $z_{17}[SR^{-1}[col:3]]$, compute $\Delta y_{18}[col:3], y_{18}[col:3]$. Then, store $\Delta z_{17}[SR^{-1}[col:3][0,1,3]]$, $z_{17}[SR^{-1}[col:3]], y_{18}[col:3][0,1]$ in H_3 indexed by $\Delta y_{18}[col:3]$, $y_{18}[col:3][2,3]$. H_3 has 2^{24} rows and on average about $2^{28}/2^{24}=2^4$ values in each row.

$\boldsymbol{H_4\{TK_{17}[0,4]\}}$: For all the 2^{20} possible values of $\Delta z_{16}[SR^{-1}[col:0][0]]$, $z_{16}[SR^{-1}[col:0]]$, compute $\Delta y_{17}[col:0][0,1,3]$, $y_{17}[col:0]$. Then, store $\Delta z_{16}[SR^{-1}[col:0][0]]$, $z_{16}[SR^{-1}[col:0]], y_{17}[col:0][0,1]$ in H_4 indexed by $\Delta y_{17}[col:0][0,1,3], y_{17}[col:0][2,3]$. H_4 has 2^{20} rows and on average about $2^{20}/2^{20}=1$ value in each row.

$\boldsymbol{H_5\{TK_{17}[2,3,6]\}}$: From the properties of the MixColumns, we have $\Delta x_{16}[0]=\Delta x_{16}[8]=\Delta x_{16}[12]=\Delta w_{15}[8]$. Therefore, for all the 2^{40} possible values for $\Delta x_{16}[8], x_{16}[8,12], \Delta w_{16}[2,7]$, $w_{16}[2,6,14], x_{17}[3,11]$, compute $w_{16}[10,15], \Delta y_{17}[2,3,6,10,11,14], y_{17}[2,3,6,10,11,14,15]$ such that $y_{17}[15]=SC([w_{16}[15]\oplus x_{17}[3])$, from the MixColumns operation. Then, store $\Delta z_{16}[SR^{-1}[col:2][0,2]], \Delta z_{16}[SR^{-1}[col:3][1,3]]$, $z_{16}[SR^{-1}[col:2]]$, $z_{16}[SR^{-1}[col:3][3]]$, $y_{17}[2,3,6]$ in H_5 indexed by $\Delta y_{17}[2,3,6,10,11,14], y_{17}[10,11,14,15]$. H_5 has 2^{40} rows and on average about $2^{40}/2^{40}=1$ value in each row.

$\boldsymbol{H_6\{TK_{17}[1,5]\}}$: For all the 2^{24} possible values of $\Delta z_{16}[SR^{-1}[col:1][0,3]], z_{16}[SR^{-1}[col:1]]$, compute $\Delta y_{17}[col:1][0,1,3], y_{17}[col:1]$. Then, store $\Delta z_{16}[SR^{-1}[col:1][0,3]]$, $z_{16}[SR^{-1}[col:1]], y_{17}[col:1][0,1]$ in H_6 indexed by $\Delta y_{17}[col:1][0,1,3], y_{17}[col:1][2,3]$. H_6 has 2^{20} rows and on average about $2^{24}/2^{20}=2^4$ values in each row.

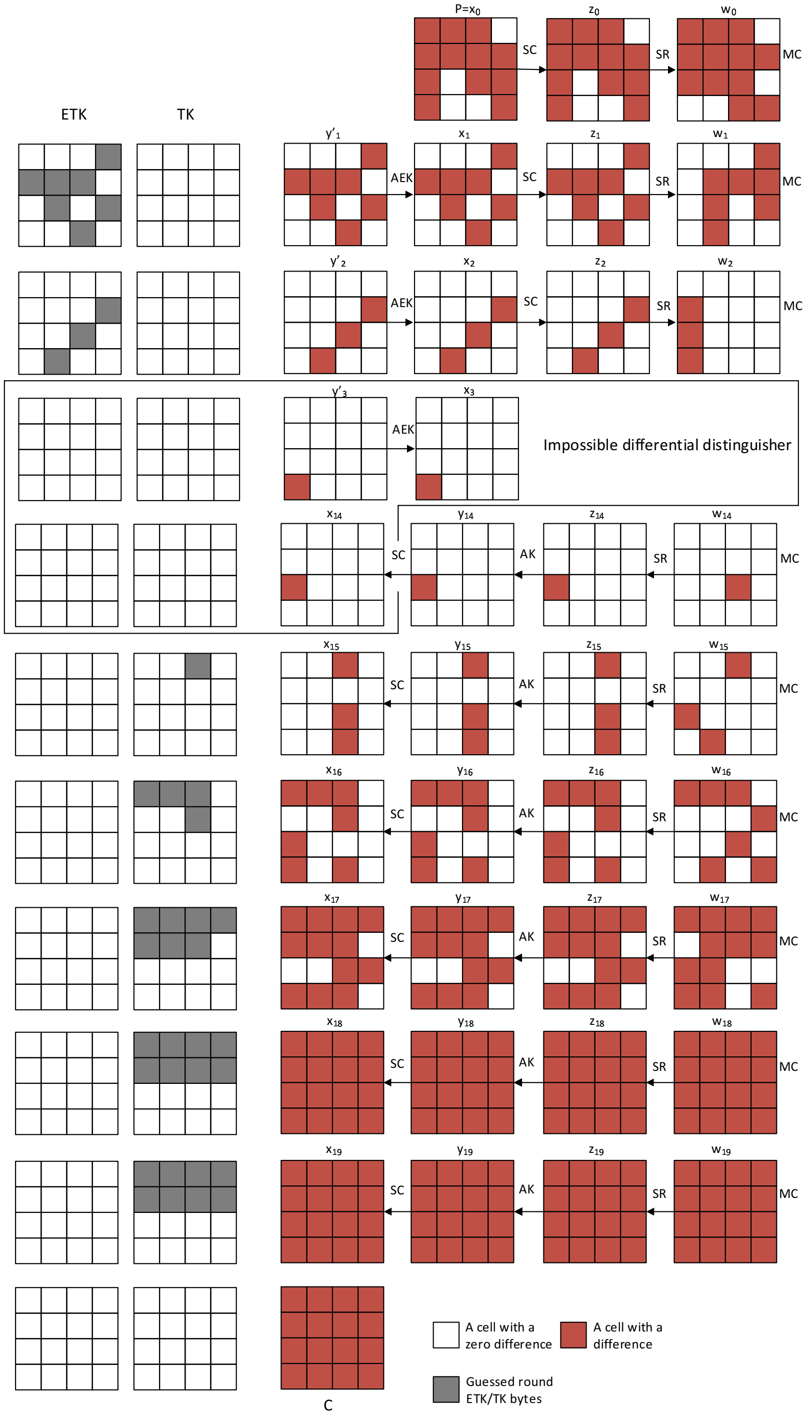


Figure 5.4: Impossible differential attack on 20-round SKINNY-n-$2n$

$\boldsymbol{H_7\{TK_{16}[0]\}}$: For all the 2^{20} possible values of $\Delta z_{15}[SR^{-1}[col:0][2]], z_{15}[SR^{-1}\ [col:0]]$, compute $\Delta y_{16}[col:0][0,2,3]$, $y_{16}[col:0]$. Then, store $\Delta z_{15}[SR^{-1}[col:0][2]]$, $z_{15}[SR^{-1}[col:0]], y_{16}[col:0][0]$ in H_7 indexed by $\Delta y_{16}[col:0][0,2,3], y_{16}[\ col:0][2,3]$. H_7 has 2^{20} rows and on average about $2^{20}/2^{20}=1$ value in each row.

$\boldsymbol{H_8\{TK_{16}[2]\}}$: For all the 2^{20} possible values of $\Delta z_{15}[SR^{-1}[col:2][0]], z_{15}[SR^{-1}\ [col:2]]$, compute $\Delta y_{16}[col:2][0,1,3]$, $y_{16}[col:2]$. Then, store $\Delta z_{15}[SR^{-1}[col:2][0]]$, $z_{15}[SR^{-1}[col:2]], y_{16}[col:2][0,1]$ in H_8 indexed by $\Delta y_{16}[col:2][0,1,3], y_{16}[\ col:2][2,3]$. H_8 has 2^{20} rows and on average about $2^{20}/2^{20}=1$ value in each row.

$\boldsymbol{H_9\{TK_{15}[2]\}}$: From the properties of the MixColumns, we have $\Delta x_{15}[2]=\Delta x_{15}[10]=\Delta x_{15}[14]=\Delta w_{14}[10]$. Therefore, for all the 2^4 possible differences for $\Delta x_{15}[2,10]$, 2^8 possible values of $x_{15}[2,10]$ and 2^4 possible values of $TK_{15}[2]$, compute $\Delta z_{15}[2\ ,10], z_{15}[2,10]$. Then, store $\Delta z_{15}[2]$ in H_9 indexed by $\Delta z_{15}[2,10], z_{15}[2,10]$, $TK_{15}[2]$. H_9 has 2^{20} rows and on average about $2^{16}/2^{20}=2^{-4}$ values in each row.

$\boldsymbol{H_{10}\{ETK_1[4,11,14]\}}$: For all the 2^{12} possible differences of $\Delta w_1[5,9,13]$, we have only 2^4 valid differences that have exactly one difference in $\Delta y_2^{'}[13]$ and 3 zero differences in $\Delta y_2^{'}[1,5,9]$. Therefore, for all the 2^4 possible differences of $\Delta w_1[5,9,13]$, 2^{12} possible values of $w_1[5,9,13]$ and 2^8 possible values of $ETK_1[4,14]$, compute $\Delta y_1^{'}[4,14]$, $y_1^{'}[4,14], \Delta x_1[11], x_1[11]$. Then, store $\Delta w_1[5,9,\ 13], w_1[5,9,13], x_1[11]$ in H_{10} indexed by $\Delta y_1^{'}[4,14], y_1^{'}[4,14], \Delta x_1[11]$, $ETK_1[4,14]$. H_{10} has 2^{28} rows and on average about $2^{24}/2^{28}=2^{-4}$ values in each row.

$\boldsymbol{H_{11}\{ETK_1[3,6,9]\}}$: For all the 2^{12} possible differences of $\Delta w_1[3,7,11]$, we have only 2^4 valid differences that have exactly one difference in $\Delta y_2^{'}[7]$ and 3 zero differences in $\Delta y_2^{'}[3,11,15]$. Therefore, for all the 2^4 possible differences of $\Delta w_1[3,7,11]$, 2^{12} possible values of $w_1[3,7,11]$ and 2^4 possible values of $ETK_1[6]$, compute $\Delta y_1^{'}[6], y_1^{'}[6], \Delta x_1[3\ ,9], x_1[3,9]$. Then, store $\Delta w_1[3,7,11], w_1[3,7,11], x_1[\ 3,9]$ in H_{11} indexed by $\Delta x_1[3,9]\ , \Delta y_1^{'}[6], y_1^{'}[6]$, $ETK_1[6]$. H_{11} has 2^{20} rows and on average about $2^{20}/2^{20}=1$ value in each row.

$\boldsymbol{H_{12}\{TK_{16}[1]\}}$: For all the 2^8 possible values of $\Delta x_{16}[1], x_{16}[1]$, compute $\Delta y_{16}[1]\ , y_{16}[1]$.Then, store $y_{16}[1]$ in H_{12} indexed by $\Delta y_{16}[1]$. H_{12} has 2^4 rows and on average about $2^8/2^4=2^4$ values in each row.

$\boldsymbol{H_{13}\{ETK_1[1,5]\}}$: For all the 2^{16} possible values of $\Delta w_1[6], w_1[1,6], ETK_1[1,5]$ ($ETK_1[1]=ETK_1[5]$, see Appendix A), compute $\Delta y_1^{'}[5], y_1^{'}[1,5]$. Then, store $\Delta w_1[6], w_1\ [1,6]$ in H_{13} indexed by $\Delta y_1^{'}[5], y_1^{'}[1,5]$, $ETK_1\ [1]$. H_{13} has 2^{16} rows and on average about $2^{16}/2^{16}=1$ value in each row.

$\boldsymbol{H_{14}\{ETK_2[7,10,13]\}}$: From the properties of the MixColumns, we have $\Delta w_2[4] = \Delta w_2[8] = \Delta w_2[12] = \Delta y_3^{'}[12]$. Therefore, for all the 2^4 possible differences for $\Delta w_2[4,8,12]$, 2^{12} possible values of $w_2[4,8,12]$ and 2^{12} possible values of $ETK_2[7,10,13]$, compute $\Delta y_2^{'}[7,10,13], y_2^{'}[7,10,13]$. Then, store $\Delta y_2^{'}[10]$ in H_{14} indexed by $\Delta y_2^{'}[7,10\ ,13], y_2^{'}[7,13]$,$ETK_2[7,10,13]$. H_{14} has 2^{32} rows and on average about $2^{28}/2^{32} = 2^{-4}$ value in each row.

$\boldsymbol{H^*}$: For all the 2^{32} possible values of $\Delta z_i[SR^{-1}[col : j]], z_i[SR^{-1}[col : j]]$, compute $\Delta y_{i+1}[col : j], y_{i+1}[col : j]$. Then, store $\Delta z_i[SR^{-1}[col : j]], z_i[SR^{-1}[col : j]], y_{i+1}[col : j][0,1]$ in H^* indexed by $\Delta y_{i+1}[col : j], y_{i+1}[col : j][2,3]$. H^* has 2^{24} rows and on average about $2^{32}/2^{24} = 2^8$ values in each row.

Instead of guessing the tweakey nibbles involved in the analysis rounds as in the general approach of impossible differential attacks, we use the above mentioned pre-computation tables to deduce the tweakey nibbles that lead a specific pair of plaintext/ciphertext to the impossible differential and thus should be excluded. The details of our attack are as follows:

1. Generate 2^m structures as described above. Therefore, we have 2^{m+55} pairs of messages generated using 2^{m+28} messages. Then, ask the encryption oracle for their corresponding ciphertexts and decrypt them partially over MC^{-1}, SR^{-1} to compute z_{19}.

2. Determine the number of possible values of $TK_{19}[0 : 7]$ that satisfy the last round by performing the following steps for all the message pairs:

   (a) Access H^* for $i = 18, j = 0$ and compute $TK_{19}[0,4]$ such that $TK_{19}[0,4] = y_{19}[0,4] \oplus z_{19}[0,4]$[1]. Therefore, we have 2^8 possible tweakeys for $TK_{19}[0,4]$.

   (b) Access H^* for $i = 18, j = 1$ and compute $TK_{19}[1,5]$ such that $TK_{19}[1,5] = y_{19}[1,5] \oplus z_{19}[1,5]$. Therefore, we have $2^{8+8=16}$ possible tweakeys for $TK_{19}[0\ ,1,4,5]$.

   (c) Access H^* for $i = 18, j = 2$ and compute $TK_{19}[2,6]$ such that $TK_{19}[2,6] = y_{19}[2,6] \oplus z_{19}[2,6]$. Therefore, we have $2^{16+8=24}$ possible tweakeys for $TK_{19}[0\ ,1,2,4,5,6]$.

   (d) Access H^* for $i = 18, j = 3$ and compute $TK_{19}[3,7]$ such that $TK_{19}[3,7] = y_{19}[3,7] \oplus z_{19}[3,7]$. Therefore, we have $2^{24+8=32}$ possible tweakeys for $TK_{19}[0 : 7]$.

3. Determine the number of possible values of $TK_{18}[0 : 7]$ that satisfy the next to last round by performing the following steps for all the message pairs and remaining tweakeys that satisfy the path until now:

[1] $TK_{19}[0,4] = y_{19}[0,4] \oplus z_{19}[0,4]$ means that $TK_{19}[0] = y_{19}[0] \oplus z_{19}[0]$, $TK_{19}[4] = y_{19}[4] \oplus z_{19}[4]$.

(a) Access H_1 and compute $TK_{18}[2,6]$ such that $TK_{18}[2,6] = y_{18}[2,6] \oplus z_{18}[2,6]$. Therefore, we have 2^{32} possible tweakeys for $TK_{19}[0:7]$, $TK_{18}[2\ ,6]$.

(b) Access H_2 and compute $TK_{18}[0,4]$ such that $TK_{18}[0,4] = y_{18}[0,4] \oplus z_{18}[0,4]$. Therefore, we have $2^{32+4=36}$ possible tweakeys for $TK_{19}[0:7], TK_{18}[0,2,4,6]$.

(c) Access H_3 and compute $TK_{18}[3,7]$ such that $TK_{18}[3,7] = y_{18}[3,7] \oplus z_{18}[3,7]$. Therefore, we have $2^{36+4=40}$ possible tweakeys for $TK_{19}[0:7], TK_{18}[0,2,3,4,\ 6,7]$.

(d) Access H^* for $i = 17, j = 1$ and compute $TK_{18}[1,5]$ such that $TK_{18}[1,5] = y_{18}[1,5] \oplus z_{18}[1,5]$. Therefore, we have $2^{40+8=48}$ possible tweakeys for $TK_{19}[0:7], TK_{18}[0:7]$.

4. Determine the number of possible values of $TK_{17}[0:6]$ that satisfy the eighteenth round by performing the following steps for all the message pairs and remaining tweakeys that satisfy the path until now:

   (a) Access H_4 and compute $TK_{17}[0,4]$ such that $TK_{17}[0,4] = y_{17}[0,4] \oplus z_{17}[0,4]$. Therefore, we have 2^{48} possible tweakeys for $TK_{19}[0:7]$, $TK_{18}[0:7]$, $TK_{17}[0,\ 4]$.

   (b) Access H_5 and compute $TK_{17}[2,3,6]$ such that $TK_{17}[2,3,6] = y_{17}[2,3,6] \oplus z_{17}[2,3,6]$. Therefore, we have 2^{48} possible tweakeys for $TK_{19}[0:7]$, $TK_{18}[0:7]$, $TK_{17}[0,2,3,4,6]$.

   (c) Access H_6 and compute $TK_{17}[1,5]$ such that $TK_{17}[1,5] = y_{17}[1,5] \oplus z_{17}[1,5]$. Therefore, we have $2^{48+4=52}$ possible tweakeys for $TK_{19}[0:7]$, $TK_{18}[0:7]$, $TK_{17}[0:6]$.

5. Determine the number of possible values of $TK_{16}[0,2]$ that satisfy the seventeenth round by performing the following steps for all the message pairs and remaining tweakeys that satisfy the path until now:

   (a) Access H_7 and compute $TK_{16}[0]$ such that $TK_{16}[0] = y_{16}[0] \oplus z_{16}[0]$. Therefore, we have 2^{52} possible tweakeys for $TK_{19}[0:7]$, $TK_{18}[0:7]$, $TK_{17}[0:6]$, $TK_{16}[0]$.

   (b) Access H_8 and compute $TK_{16}[2]$ such that $TK_{16}[2] = y_{16}[2] \oplus z_{16}[2]$. Therefore, we have 2^{52} possible tweakeys for $TK_{19}[0:7]$, $TK_{18}[0:7]$, $TK_{17}[0:6]$, $TK_{16}[0,2]$[2].

6. The knowledge of $TK_{19}[6]$ and $TK_{17}[4]$ enables us to deduce $TK_{15}[2]$ (see Appendix A). Hence, we determine the number of possible tweakey values that satisfy the sixteenth round by performing the following steps for all the message pairs and remaining tweakeys that satisfy the path until now:

   (a) Access H_9; and we will find 2^{-4} possible values in each row, i.e., we have 4-bit filter on the remaining tweakeys. Therefore, we have $2^{52-4=48}$ possible tweakeys for $TK_{19}[0:7]$, $TK_{18}[0:7]$, $TK_{17}[0:6]$, $TK_{16}[0,2]$ $TK_{15}[2]$.

[2] Note that instead of having $TK_{16}[6]$ that lead to the impossible differential distinguisher, we have $x_{16}[6]$ that result in the same impossible differential distinguisher.

7. The knowledge of $TK_{18}[2,4]$ and $TK_{16}[0,2]$ enables us to deduce $ETK_1[4,6,\ 14]$ [3] (see Appendix A). Hence, we determine the number of possible values for $ETK_1[3,9,11]$ that satisfy the second round by performing the following steps for all the message pairs and remaining tweakeys that satisfy the path until now:

   (a) Access H_{10} and compute $ETK_1[11]$ such that $ETK_1[11] = y_1^{'}[11] \oplus x_1[11]$; we will find 2^{-4} possible values in each row, i.e., we have 4-bit filter on the remaining tweakeys. Therefore, we have $2^{48-4=44}$ possible tweakeys for $TK_{19}[0:7]$, $TK_{18}[0:7]$, $TK_{17}[0:6]$, $TK_{16}[0,2]$, $TK_{15}[2]$, $ETK_1[4,6,11,\ 14]$.

   (b) Access H_{11} and compute $ETK_1[3,9]$ such that $ETK_1[3,9] = y_1^{'}[3,9] \oplus x_1[3,9]$. Therefore, we have 2^{44} possible tweakeys for $TK_{19}[0:7]$,$TK_{18}[0:7]$, $TK_{17}[0:6]$, $TK_{16}[0,2]$, $TK_{15}[2]$, $ETK_1[3,4,6,9,11,14]$.

8. Determine the number of possible values for $TK_{16}[1]$ that satisfy the seventeenth round by performing the following steps for all the message pairs and remaining tweakeys that satisfy the path until now:

   (a) Access H_{12} and compute $TK_{16}[1]$ such that $TK_{16} = y_{16}[1] \oplus z_{16}[1]$. Therefore, we have $2^{44+4=48}$ possible tweakeys for $TK_{19}[0:7]$,$TK_{18}[0:7]$, $TK_{17}[0:6]$, $TK_{16}[0,1,2]$, $TK_{15}[2]$, $ETK_1[3,4,6,9,11,14]$.

9. The knowledge of $TK_{18}[0]$ and $TK_{16}[1]$ enables us to deduce $ETK_1[1,5]$[4] (see Appendix A). Hence, we determine the number of possible tweakey values that satisfy the second round by performing the following steps for all the message pairs and remaining tweakeys that satisfy the path until now:

   (a) Access H_{13} and we will find 1 possible value in each row. Therefore, we have 2^{48} possible tweakeys for $TK_{19}[0:7]$, $TK_{18}[0:7]$, $TK_{17}[0:6]$, $TK_{16}[0,1,2]$, $TK_{15}[2]$, $ETK_1[1,3,4,5,6,9,11,14]$,.

10. The knowledge of $TK_{19}[0,3,7]$ and $TK_{17}[1,3,5]$ enables us to deduce $ETK_2[7,\ 10,13]$ (see Appendix A). Hence, we determine the number of possible tweakey values that satisfy the third round by performing the following steps for all the message pairs and remaining tweakeys that satisfy the path until now:

    (a) Access H_{14} and we will find 2^{-4} possible values in each row. Therefore, we have $2^{48-4=44}$ possible tweakeys for $TK_{19}[0:7]$, $TK_{18}[0:7]$, $TK_{17}[0:6]$, $TK_{16}[0,1,2]$, $TK_{15}[2]$, $ETK_1[1,3,4,5,6,9,11,14]$, $ETK_2[7,10,13]$.

[3] Note that $ETK_1[6] = ETK_1[14]$.

[4] Note that $ETK_1[1] = ETK_1[5]$.

Attack Complexity. As depicted in Figure 5.4, we have 38 tweakey nibbles that are involved in the analysis rounds. Thanks to the tweakey schedule, these 38 nibbles take only 2^{116} possible values (see Appendix A). For each of the 2^{m+55} message pairs, we remove, on average, 2^{44} out of 2^{116} possible values of these tweakey nibbles. Therefore, the probability that a wrong tweakey is not discarded with one pair is $1-2^{44-116} = 1-2^{-72}$. Hence, after processing all the 2^{m+55} pairs, we have $2^{116}(1-2^{-72})^{2^{m+55}} \approx 2^{116} \times (e^{-1})^{2^{m+55-72}} \approx 2^{116} \times 2^{-1.4\times 2^{m-17}}$ remaining candidates for 116-bit of the tweakey. In order to determine the optimal value of m that leads to the best computational complexity, we evaluate the computational complexity of the attack as a function of m, as illustrated in Table 5.4. Similar to AES [54], the SKINNY round function can be implemented using 16 table lookups. As seen from Table 5.4, steps 5(a), 5(b) and 6(a) dominate the time complexity of the attack, and hence in order to optimize the time complexity of the attack we choose $m = 19.69$. Consequently, we have 2^{107} remaining tweakey candidates for the 116-bit of the tweakey. Therefore, the tweakey can be recovered by exhaustively searching the 2^{107} remaining tweakey candidates with 2^{12} remaining tweakey bits, that are not involved in the attack, using 2 plaintext/ciphertext pairs. Therefore, the total time complexity of the attack is $2 \times 2^{107} \times 2^{12} + 2^{120.15} = 2^{121.08}$ encryptions. The data complexity of the attack can be determined from step 1 in which we generate $2^{m=19.69}$ structures. Hence, the data complexity of the attack is $2^{19.69+28=47.69}$ chosen plaintexts. The memory complexity of the attack is dominated by the memory that is required to store $2^{m+55=74.69}$ pairs to exclude the wrong tweakeys, hence, it is $2^{74.69}$.

5.4.2 Impossible Differential Key-recovery Attack on SKINNY-128-256

The only difference between SKINNY-64-128 and SKINNY-128-256 is the tweakey schedule, more precisely, the LFSR operation. The above attack on SKINNY-64-128 can be applied on SKINNY-128-256 while only considering that the cell size $s = 8$. Therefore, one structure can generate 2^{111} pairs with 2^{56} chosen plaintexts. According to the tweakey schedule, the 38 bytes involved in the attack have 2^{232} possible values (see Appendix B). In this attack, we exclude, on overage, 2^{88} out of 2^{232} possible values of the involved tweakey bytes for every message pair. Hence, the probability that one wrong tweakey is not discarded is $1-2^{88-232} = 1-2^{-144}$. Therefore, we have $2^{232} \times (1-2^{-144})^{2^{m+111}} \approx 2^{232} \times (e^{-1})^{2^{m+111-144}} \approx 2^{232} \times 2^{-1.4\times 2^{m-33}}$ remaining candidates for 232-bit of the tweakey bytes, after processing all the message pairs. In order to optimize the time complexity of the attack, we choose $m = 36.1$. Consequently, we have 2^{220} remaining candidates for 232-bit of the tweakey, and hence the tweakey can be recovered by exhaustively searching the remaining candidates with 2^{24} possible values, for the 24 bits of the tweakey that are not involved in the attack, using 2 plaintext/ciphertext pairs. Therefore, the total time complexity of the attack is

Table 5.4: Time complexity of the different steps of the attack on 20-round SKINNY-64-128, where NT denotes the number of tweakeys to be excluded

Step	Time Complexity (in 20-round encryptions)	NT	$m = 19.69$
1	2^{m+28}	-	$2^{47.69}$
2(a)	$2^{m+55} \times \frac{1}{16 \times 20} \approx 2^{m+46.68}$	2^{8}	$2^{66.37}$
2(b)	$2^{m+55} \times 2^{8} \times \frac{1}{16 \times 20} \approx 2^{m+54.68}$	2^{16}	$2^{74.37}$
2(c)	$2^{m+55} \times 2^{16} \times \frac{1}{16 \times 20} \approx 2^{m+62.68}$	2^{24}	$2^{82.37}$
2(d)	$2^{m+55} \times 2^{24} \times \frac{1}{16 \times 20} \approx 2^{m+70.68}$	2^{32}	$2^{90.37}$
3(a)	$2^{m+55} \times 2^{32} \times \frac{1}{16 \times 20} \approx 2^{m+78.68}$	2^{32}	$2^{98.37}$
3(b)	$2^{m+55} \times 2^{32} \times \frac{1}{16 \times 20} \approx 2^{m+78.68}$	2^{36}	$2^{98.37}$
3(c)	$2^{m+55} \times 2^{36} \times \frac{1}{16 \times 20} \approx 2^{m+82.68}$	2^{40}	$2^{102.37}$
3(d)	$2^{m+55} \times 2^{40} \times \frac{1}{16 \times 20} \approx 2^{m+86.68}$	2^{48}	$2^{106.37}$
4(a)	$2^{m+55} \times 2^{48} \times \frac{1}{16 \times 20} \approx 2^{m+94.68}$	2^{48}	$2^{114.37}$
4(b)	$2^{m+55} \times 2^{48} \times \frac{2}{16 \times 20} \approx 2^{m+95.68}$	2^{48}	$2^{115.37}$
4(c)	$2^{m+55} \times 2^{48} \times \frac{1}{16 \times 20} \approx 2^{m+94.68}$	2^{52}	$2^{114.37}$
5(a)	$2^{m+55} \times 2^{52} \times \frac{1}{16 \times 20} \approx 2^{m+98.68}$	2^{52}	$2^{118.37}$
5(b)	$2^{m+55} \times 2^{52} \times \frac{1}{16 \times 20} \approx 2^{m+98.68}$	2^{52}	$2^{118.37}$
6(a)	$2^{m+55} \times 2^{52} \times \frac{1}{16 \times 20} \approx 2^{m+98.68}$	2^{48}	$2^{118.37}$
7(a)	$2^{m+55} \times 2^{48} \times \frac{1}{16 \times 20} \approx 2^{m+94.68}$	2^{44}	$2^{114.37}$
7(b)	$2^{m+55} \times 2^{44} \times \frac{1}{16 \times 20} \approx 2^{m+90.68}$	2^{44}	$2^{110.37}$
8(a)	$2^{m+55} \times 2^{44} \times \frac{1}{16 \times 20} \approx 2^{m+90.68}$	2^{48}	$2^{110.37}$
9(a)	$2^{m+55} \times 2^{48} \times \frac{1}{16 \times 20} \approx 2^{m+94.68}$	2^{48}	$2^{114.37}$
10(a)	$2^{m+55} \times 2^{48} \times \frac{1}{16 \times 20} \approx 2^{m+94.68}$	2^{44}	$2^{114.37}$

$2 \times 2^{220} \times 2^{24} + 2^{36.1+111} \times 2^{104} \times \frac{3}{16\times20}$[5]$= 2^{245} + 2^{244.36} = 2^{245.72}$. The data complexity of the attack is $2^{m+56=92.1}$ chosen plaintexts; and the memory complexity is dominated by storing $2^{m+111=147.1}$ message pairs.

5.5 Impossible Differential Key-recovery Attack on 18-round SKINNY-n-n

The only difference between SKINNY-64-64 and SKINNY-128-128 is the cell size s, where $s = 4$ (resp. $s = 8$) in case of SKINNY-64-64 (resp. SKINNY-128-128). Therefore, we present the steps of the two attacks concurrently as a function of s. This attack is applicable to the first 18 rounds of the 20-round attack on SKINNY-n-$2n$, i.e., the ciphertext $c = x_{18}$. Therefore, we use the same steps used in the previous attack from step 4 to the end and the same precomputation tables from H_4 to the end with the following modifications:

- Each structure can generate $2^{7\times s} \times 2^{7\times s-1} = 2^{14\times s-1}$ with $2^{7\times s}$ chosen plaintexts. Then, to apply the attack we take 2^m structures to generate $2^{m+14\times s-1}$ pairs, but we have 4 s-bit filter in the transition over MC^{-1} from the ciphertext to w_{17}. Therefore, we have $2^{m+14\times s-1-4\times s=m+10\times s-1}$ remaining pairs to launch the attack.

- The number of rows and entries in each table will be represented as a function of s. For example, H_6 has $2^{5\times s}$ rows; and in each row, we have 2^s entries.

- The modifications of the number of tweakeys to be excluded from step 4 to the end are presented in Table 5.5.

- The relation of the tweakey cells can be found in Appendix C.

Attack Complexity. We have 22 tweakey cells that are involved in the analysis rounds where these 22 tweakey cells have only $2^{13\times s}$ possible values (see Appendix C). The probability that one wrong tweakey is not discarded with one pair is $1 - 2^{-s-13\times s} = 1 - 2^{-14\times s}$. Hence, after processing all the $2^{m+10\times s-1}$ pairs, we have $2^{13\times s}(1 - 2^{-14\times s})^{2^{m+10\times s-1}} \approx 2^{13\times s} \times (e^{-1})^{2^{m+10\times s-1-14\times s}} \approx 2^{13\times s} \times 2^{-1.4\times 2^{m-4\times s-1}}$ remaining candidates for $13 \times s$-bit of the tweakey. Steps 5(a), 5(b) and 6(a) dominate the time complexity of the attack, as seen from Table 5.5, and hence in order to optimize the time complexity of the attack we choose $m = 19.52$ (resp. $m = 36.42$) in case of SKINNY-64-64 (resp. SKINNY-128-128). Consequently, we have 2^{44} (resp. 2^{89}) remaining tweakey candidates for the 52-bit (resp. 104-bit) of the tweakey. Therefore, the tweakey can be recovered by exhaustively searching the 2^{44} (resp. 2^{89}) remaining tweakey candidates with 2^{12} (resp. 2^{24}) for the other tweakey bits, that are not involved in

[5] The second term is computed from step 5(a), 5(b) and 6(a).

the attack, using 1 plaintext/ciphertext pair. Therefore, the total time complexity of the attack is $2^{44} \times 2^{12} + 2^{56.14} = 2^{57.1}$ (resp. $2^{89} \times 2^{24} + 2^{116.84} = 2^{116.94}$) encryptions in case of SKINNY-64-64 (resp. SKINNY-128-128). The data complexity of the attack can be determined from step 1 in which we generate $2^{m=19.52}$ (resp. $2^{m=36.42}$) structures. Hence, the data complexity of the attack is $2^{19.52+28=47.52}$ (resp. $2^{36.42+56=92.42}$) chosen plaintexts in case of SKINNY-64-64 (resp. SKINNY-128-128). The memory complexity is dominated by the memory required to store the $2^{58.52}$ (resp. $2^{115.42}$) pairs after the ciphertext filtration and is estimated to be $2^{58.52}$ (resp. $2^{115.42}$) in case of SKINNY-64-64 (resp. SKINNY-128-128).

Table 5.5: Time complexity of the different steps of the attack on 18-round SKINNY-64-64 and SKINNY-128-128, where NT denotes the number of tweakeys to be excluded

Step	Time Complexity (in 18-round encryptions)	NT	$s = 4, m = 19.52$	$s = 8, m = 36.42$
1	$2^{m+7\times s}$	-	$2^{47.52}$	$2^{92.42}$
4(a)	$2^{m+10\times s-1} \times \frac{1}{16 \times 18} \approx 2^{m+10\times s-9.17}$	1	$2^{50.35}$	$2^{107.25}$
4(b)	$2^{m+10\times s-1} \times \frac{2}{16 \times 18} \approx 2^{m+10\times s-8.17}$	1	$2^{51.35}$	$2^{108.25}$
4(c)	$2^{m+10\times s-1} \times \frac{1}{16 \times 18} \approx 2^{m+10\times s-9.17}$	2^s	$2^{50.35}$	$2^{107.25}$
5(a)	$2^{m+10\times s-1} \times 2^s \times \frac{1}{16 \times 18} \approx 2^{m+11\times s-9.17}$	2^s	$2^{54.35}$	$2^{115.25}$
5(b)	$2^{m+10\times s-1} \times 2^s \times \frac{1}{16 \times 18} \approx 2^{m+11\times s-9.17}$	2^s	$2^{54.35}$	$2^{115.25}$
6(a)	$2^{m+10\times s-1} \times 2^s \times \frac{1}{16 \times 18} \approx 2^{m+11\times s-9.17}$	1	$2^{54.35}$	$2^{115.25}$
7(a)	$2^{m+10\times s-1} \times \frac{1}{16 \times 18} \approx 2^{m+10\times s-9.17}$	2^{-s}	$2^{50.35}$	$2^{107.25}$
7(b)	$2^{m+10\times s-1} \times 2^{-s} \times \frac{1}{16 \times 18} \approx 2^{m+9\times s-9.17}$	2^{-s}	$2^{46.35}$	$2^{99.25}$
8(a)	$2^{m+10\times s-1} \times 2^{-s} \times \frac{1}{16 \times 18} \approx 2^{m+9\times s-9.17}$	1	$2^{46.35}$	$2^{99.25}$
9(a)	$2^{m+10\times s-1} \times \frac{1}{16 \times 18} \approx 2^{m+10\times s-9.17}$	1	$2^{50.35}$	$2^{107.25}$
10(a)	$2^{m+10\times s-1} \times \frac{1}{16 \times 18} \approx 2^{m+10\times s-9.17}$	2^{-s} [6]	$2^{50.35}$	$2^{107.25}$

5.6 Impossible Differential Key-recovery Attack on 22-round SKINNY-n-3n

SKINNY-64-192 differs from SKINNY-128-384 in the cell size s and the tweakey schedule. As the tweakey schedule does not influence the attack procedure, we present the two attacks as a function of s. The 20-round attack on SKINNY-n-$2n$ ($n = 64$ or 128) can be extended to 22-round attack on SKINNY-n-$3n$ ($n = 64$ or 128) by appending 2 rounds, i.e., the ciphertext $c = x_{22}$. Therefore, we can use the same attack procedures of SKINNY-n-$2n$ ($n = 64$ or 128)

[6] After this step, we have 2^{-s} tweakeys to be excluded for each message pair, i.e., we exclude 1 tweakey after processing 2^s pairs.

to attack SKINNY-n-$3n$ ($n = 64$ or 128) by repeating step 2 three times to extract the tweakey cells $TK_{19}[0:7]$, $TK_{20}[0:7]$, $TK_{21}[0:7]$. The details of the tweakey schedule can be found in Appendix D. Moreover, as in the previous attack on 18-round SKINNY-n-n ($n = 64$ or 128), each structure can generate $2^{7\times s} \times 2^{7\times s-1} = 2^{14\times s-1}$ with $2^{7\times s}$ chosen plaintexts. Then, we take 2^m structures to generate $2^{m+14\times s-1}$ pairs using $2^{m+7\times s}$ chosen plaintexts.

Attack Complexity. The 54 tweakey cells that are involved in the analysis rounds have only $2^{45\times s}$ possible values. The probability that a wrong tweakey is not discarded with one pair is $1 - 2^{27\times s-45\times s} = 1 - 2^{-18\times s}$. Hence, after processing all the $2^{m+14\times s-1}$ pairs, we have $2^{45\times s}(1 - 2^{-18\times s})^{2^{m+14\times s-1}} \approx 2^{45\times s} \times (e^{-1})^{2^{m+14\times s-1-18\times s}} \approx 2^{45\times s} \times 2^{-1.4\times 2^{m-4\times s-1}}$ remaining candidates for $45 \times s$-bit of the tweakey. In order to optimize the time complexity of the attack, we choose $m = 19.84$ (resp. $m = 36.22$) in case of SKINNY-64-192 (resp. SKINNY-128-384). Consequently, we have 2^{170} (resp. 2^{347}) remaining tweakey candidates for the 180-bit (resp. 360-bit) of the tweakey. Therefore, the tweakey can be recovered by exhaustively searching the 2^{170} (resp. 2^{347}) remaining tweakey candidates with 2^{12} (resp. 2^{24}) for the other tweakey bits, that are not involved in the attack, using 3 (calculated from the unicity distance) plaintext/ciphertext pairs. Therefore, the total time complexity of the attack is $3 \times 2^{170} \times 2^{12} + 2^{183.97} = 2^{184.79}$ (resp. $3 \times 2^{347} \times 2^{24} + 2^{372.35} = 2^{373.48}$) encryptions in case of SKINNY-64-192 (resp. SKINNY-128-384). The data complexity of the attack is $2^{19.84+28=47.84}$ (resp. $2^{36.22+56=92.22}$) chosen plaintexts in case of SKINNY-64-192 (resp. SKINNY-128-384). The memory complexity of the attack is $2^{74.84}$ (resp. $2^{147.22}$) in case of SKINNY-64-64 (resp. SKINNY-128-384).

5.7 Conclusion

In this chapter, we presented impossible differential attacks against reduced-round versions of all the 6 SKINNY's variants. All of these attacks use the same impossible differential distinguisher that covers 11-round. We extended this 11-round distinguisher by 7, 9 and 11 rounds to attack 18, 20 and 22 rounds of SKINNY-n-n, SKINNY-n-$2n$ and SKINNY-n-$3n$ ($n = 64$ or 128), respectively, exploiting the properties of the MixColumns operation, the simple tweakey schedule and the fact that the tweakey is only added to the first two rows of the state. The presented attacks are currently the best known ones on all the variants of SKINNY in the single-tweakey model.

Chapter 6

Cryptanalysis of Midori128

Midori is a family of SPN-based lightweight block ciphers designed to optimize the hardware energy consumption per bit during the encryption and decryption operations. At ASIACRYPT 2015, two variants of the cipher, Midori128 and Midori64, which support a 128-bit secret key and a 128/64-bit block, respectively, were proposed. Recently, a Meet-in-the-Middle attack and an invariant subspace attack were presented against Midori64 but both attacks cannot be applied to Midori128. The only published attack against Midori128 was launched using impossible differential and covers 10 rounds without the pre-whitening key. In this chapter, we investigate the security of the low energy block cipher, Midori128, against two types of attacks, namely, multiple impossible differential and truncated differential.

Firstly, by exploiting the special structure of the S-boxes and the binary linear transformation layer in Midori128, we present impossible differential distinguishers that cover 7 full rounds including the mix column operations. Then, we exploit four of these distinguishers to launch a multiple impossible differential attack against 11 rounds of the cipher with the pre-whitening and post-whitening keys.

Secondly, we present truncated and multiple differential cryptanalysis of round reduced Midori128. Our analysis utilizes the special structure of the S-boxes and binary linear transformation layer in order to minimize the number of active S-boxes. In particular, we consider differentials that contain only single bit differences in the input and output of the active S-boxes. To keep this single bit per S-box patterns after the $MixColumn$ operation, we restrict the bit differences of the output of the active S-boxes, which lie in the same column after the shuffle operation, to be in the same position. Using these restrictions, we were able to find 10-round differential which holds with probability 2^{-118}. By adding two rounds above and one round below this differential, we obtain a 13 round truncated differential and use it to perform a key recovery attack on the 13-round reduced Midori128. We also present a multiple differential attack on the 13-round Midori128.

6.1 Introduction

Over the past few years, many lightweight block ciphers such as HIGHT [68], mCrypton [94], DESL/DESXL [88], PRESENT [41], KATAN/KTANTAN [46], Piccolo [116] and PRINT-cipher [78] were proposed. On the other hand, there has been little work that focuses on determining the design choices that lead to the most energy efficient architecture. While power and energy are clearly correlated, optimizing the power consumption for block ciphers does not necessarily lead to the most energy efficient designs since a low power optimized cipher may have high latency, i.e., it takes longer to perform the encryption and decryption operations and hence the required energy increases. In other words, there is no guarantee that low power block cipher designs would lead to low energy designs and vice versa.

By identifying some design choices that are energy efficient and by choosing components specifically tailored to meet the requirements of low energy design, at ASIACRYPT 2015, Banik *et al.* [18] proposed an SPN-based lightweight block cipher, Midori. In particular, Midori is designed to optimize the hardware energy consumption per bit during the encryption and decryption operations. Two variants of the cipher, namely, Midori128 and Midori64 which support a 128-bit secret key and a 128/64-bit block, respectively, were proposed. The linear and non-linear operations of both versions were selected to optimize this objective.

The state in Midori is represented as a 4×4 matrix, where the size of each cell depends on the version of cipher, e.g., the cell size in Midori128 is 8 bits. Midori uses 4×4 almost MDS binary matrix because, compared to other MDS matrices, this almost MDS matrix is more efficient in terms of area and signal-delay. To compensate for the low branch number of the almost MDS matrix (4 as compared to 5 in the case of MDS), the designers utilized an optimal cell-permutation layer in order to improve the diffusion speed and increase the number of active S-boxes.

Recently, Midori64 has been analyzed by two different techniques. The first is a meet-in-the-middle with differential enumeration and key dependent sieving [95] which attacks 11 rounds (resp. 12 rounds) with time, memory, and data complexities of 2^{122} (resp. $2^{125.5}$) encryptions, $2^{89.2}$ (resp. 2^{106}) 64-bit blocks, and 2^{53} (resp. $2^{55.5}$) chosen plaintext. The second is an invariant subspace attack [66] against the full cipher. The latter attack proves that the security margin of Midori64 is 96 bits instead of 128 bits. Both of these attacks are not applicable to Midori128.

While the designers of Midori do not claim resistance under the related, known or chosen-key attack models [18], a related key attack against the cipher was presented in [64]. In the single-key attack model, the Midori128 has been analyzed in [51, 115]. In particular, Chen *et al.* [51] presented a 10-round impossible differential attack without the pre-whitening key utilizing a 6-round distinguisher. Later on, Sasaki and Todo [115] proposed a new tool to find impossible differential distinguishers for symmetric-key primitives, and they presented a 7-round distinguisher for Midori128 but the last round in their distinguisher contains the $SubCell$ operation only.

In the first part of this chapter, we improve the previous results of the impossible differential cryptanalysis against Midori128. More specifically, we present several impossible differential distinguishers against Midori128 that cover 7 full rounds (including the linear transformation layer of the the last round). These distinguishers exploit the structure of the S-boxes that are used in Midori128 along with the binary nature of the mix column operation. In particular, we exploit that each S-box of the 4 different 8-bit S-boxes that are used in Midori128 is composed of two 4-bit S-boxes (see Figure 6.1). Then, we choose differences that activate only one of these 4-bit S-boxes to find such distinguishers. Then, using four of these impossible differential distinguishers, we present 11-round multiple impossible differential attack against Miroi128 including the pre-whitening and post-whitening keys.

In the second part of this chapter, we present truncated differential cryptanalysis of round reduced Midori128. Our attack utilizes the following two observations. First, Midori128 uses four different 8-bit S-boxes, namely SSb_0, SSb_1, SSb_2 and SSb_3, where each one is composed of two 4-bit S-boxes Sb_1 in addition to input and output bit permutation. Consequently, in order to minimize the number of active S-boxes, we consider only single bit differences (i.e., $1, 2, 4, 8, 16, 32, 64, 128$) in the input and output of the 8-bit S-boxes. Second, given the binary nature of the almost MDS transformation, and the fact that the Hamming weight of each row is 3, it follows that the active bytes in a column after the $MixColumn$ operation have a single bit difference if and only if the active bytes in the input column are all equal and each one has a single active bit. Hence, to maintain the pattern of single bit differences after the $MixColumn$ operation, we restrict the bit differences of the output of the active S-boxes, which lie in the same column after the shuffle operation, to be in the same position. Based on these observations, we are able to find a 10-round differential that holds with probability 2^{-118}. Then, we added two rounds above and one round below this 10-round differential to obtain a 13-round truncated differential [80] that holds with probability 2^{-230}. Using this truncated differential, we can recover the master key of the 13-round reduced cipher with time and data complexities of 2^{119} encryptions, and 2^{119} chosen plaintext, respectively. We also present a multiple differential attack [47] on the 13-round reduced cipher with time and data complexities of $2^{125.7}$ encryptions and $2^{115.7}$ chosen plaintext, respectively.

The rest of the chapter is organized as follows. In section 6.2, we provide a brief description of Midori128. In section 6.3, we present our 7 rounds impossible differential distinguishers against Midori128 and give the details of our 11 rounds multiple impossible differential attack on Midori128. In section 6.4, we describe and analyze the algorithm we use to efficiently search for long differentials with small number of active S-boxes and give the details of our truncated differential attack on Midori128 reduced to 13 rounds in addition to our multiple differential attack. Finally, the chapter is concluded in section 6.5.

6.2 Specifications of Midori128

Midori128 can be considered as a variant of Substitution Permutation Networks (SPNs). The state in Midori128 is represented as a 4 × 4 array of bytes as follow:

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

Midori128 iterates over 20 rounds. Each round, except the last one, has 3 layers: S-layer (*SubCell*) which maps $\{0,1\}^{128} \to \{0,1\}^{128}$, P-layer (*ShuffleCell* and *MixColumn*) which maps $\{0,1\}^{128} \to \{0,1\}^{128}$ and a key-addition layer (*KeyAdd*) which maps $\{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$. The last round contains only the S-layer. Moreover, before the first and after the last rounds, prewhitening and postwhitening are performed using WK. In what follows, we show how these operations update the 128-bit state S:

- *SubCell*: A nonlinear layer applies one of four 8-bit S-boxes, namely SSb_0, SSb_1, SSb_2, and SSb_3, on each byte of the state S in parallel, where $s_i \leftarrow SSb_{(i \bmod 4)}\,[s_i]$, $0 \le i \le 15$. As shown in Figure 6.1, each 8-bit S-box SSb_i is composed of 8-bit input permutation, p_i, two 4-bit S-boxes (Sb_1, see Table 6.1) and 8-bit output permutation, p_i^{-1}.

- *ShuffleCell*: The bytes of the state S is permuted as follow: $(s_0, s_1, \cdots, s_{15}) \leftarrow (s_0, s_{10}, s_5, s_{15}, s_{14}, s_4, s_{11}, s_1, s_9, s_3, s_{12}, s_6, s_7, s_{13}, s_2, s_8)$

- *MixColumn*: Each column in the internal state is multiplied by a binary matrix M, where

Table 6.1: 4-bit bijective S-box Sb_1 in hexadecimal form [18]

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$Sb_1[x]$	1	0	5	3	e	2	f	7	d	a	9	b	c	8	4	6

$$M = M^{-1} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Hence, the internal state is updated as follows:

$$(s_i, s_{i+1}, s_{i+2}, s_{i+3})^t \leftarrow M(s_i, s_{i+1}, s_{i+2}, s_{i+3})^t, i = 0, 4, 8, 12.$$

- *KeyAdd*: Where 128-bit round key RK_i is XORed with the state S.

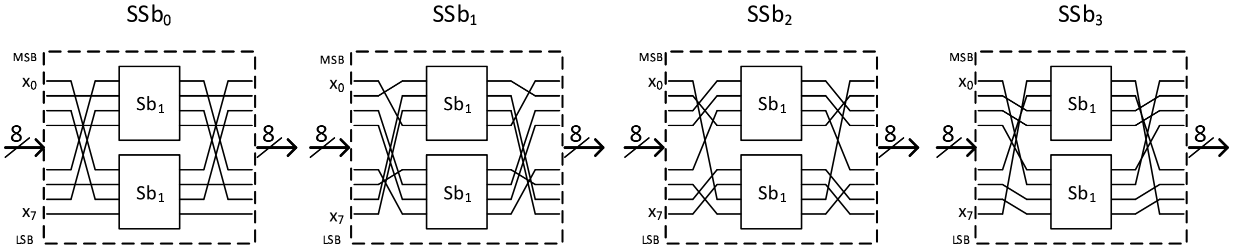


Figure 6.1: SSb_0, SSb_1, SSb_2, and SSb_3 [18]

The data encryption procedure of Midori128 is illustrated in Algorithm 7 where $R = 20$ denotes the number of rounds. The prewhitening and postwhitening key WK in Midori128 are equal to the master key K, the rounds keys $RK_i = K \oplus \beta_i$, $0 \leq i \leq 18$, where β_i is a constant, X is the plaintext, and Y is the ciphertext. Throughout our analysis, we measure the time complexity of our attack in terms of the equivalent number of reduced-round Midori128 encryptions. For further details about the design rational of the cipher, the reader is referred to [18].

The following notations are used throughout the rest of the chapter:

Algorithm 7: Data Encryption Algorithm [18]

```
Data: $X, WK, RK_0, ..., RK_{R-2}$
Result: $Y$
$S \leftarrow KeyAdd(X, WK)$;
for $i \leftarrow 0$ to $R-2$ do
    $S \leftarrow SubCell(S)$;
    $S \leftarrow ShuffleCell(S)$;
    $S \leftarrow MixColumn(S)$;
    $S \leftarrow KeyAdd(S, RK_i)$;
$S \leftarrow SubCell(S)$;
$Y \leftarrow KeyAdd(S,\ WK)$;
```

- a^t: Transposition of the vector or the matrix a.

- K: The master key.

- RK_i: The 128-bit round key used in round i.

- WK: The 128-bit whitening key.

- x_i: The 128-bit input to the *SubCell* operation at round i.

- y_i: The 128-bit input to the *ShuffleCell* operation at round i.

- z_i: The 128-bit input to the *MixColumn* operation at round i.

- w_i: The 128-bit input to the *KeyAdd* operation at round i.

- $x_i[j]$: The j^{th} byte of x_i, where $0 \leq j < 16$.

- $x_i[j_t](resp.x_i[j_b])$: The top (resp. bottom) 4-bit of $p_1(x_i[j])$.

- $x_i[j, \cdots, l]$: The bytes $j, \cdots, l$ of x_i.

- $x_i[j \cdots l]$: The bytes from j to l of x_i, where $j < l$.

- $\Delta x_i, \Delta x_i[j]$: The difference at state x_i and byte $x_i[j]$, respectively.

Throughout our analysis, we utilize the differntial property of the S-box (see Proposition 1).

6.3 Improved Multiple Impossible Differential Cryptanalysis of Midori128

6.3.1 7-round Impossible Differential Distinguishers of Midori128

All of our 7-round distinguishers begin with one active byte $w_1[i]$, $0 \leq i \leq 15$, such that only the top (resp. bottom) 4 bits of $p_{i\, mod\, 4}(w_1[i])$ are active and ends with two active bytes $x_9[i,j](i,j \in \{4l, 4l+1, 4l+2, 4l+3\}, 0 \leq l < 4, i \neq j)$ where only the top (resp. bottom) 4 bits of $p_{i\, mod\, 4}(x_9[i])$ and $p_{i\, mod\, 4}(x_9[j])$ are active. It can be verified that there are 24 such impossible differential distinguishers out of the possible $2 \times 16 \times 4 \times \binom{4}{2} = 768$, where we have 16 possible positions for $w_1[i]$, and for $x_9[i,j]$ we have 4 columns where in each column we have $\binom{4}{2}$ combinations for the positions of i, j, and in each one of these patterns we can activate the top or bottom 4-bit S-box. One of these distinguishers is illustrated in Figure 6.2 and is based on the following propositions.

Proposition 7 *Let $\Delta = xx0000xx$, where 0 and x denote the inactive and active/inactive bits, respectively, and at least one of the x bits should be active. The probability of $\Delta \xrightarrow{SSb_1} \Delta = 1$.*

Proof. This property follows from the structural properties of the S-box, as shown in Figure 6.3. As depicted in this figure, after applying the input bit permutation p_1, the input of the top 4-bit S-box Sb_1 has the difference $xxxx$ (and hence it is active) while the input of the bottom 4-bit S-box Sb_1 has the difference 0000 (and hence it is inactive). Since the S-box Sb_1 is bijective, a non-zero input difference to the S-box, implies a non-zero output difference. Therefore, the outputs of the top and bottom 4-bit S-boxes Sb_1 have the difference $xxxx$ and 0000, respectively, with probability one. Then, applying the bit permutation p_1^{-1} makes the output of the 8-bit S-box SSb_1 has the difference pattern Δ. This property is also applicable to SSb_1^{-1}.

Proposition 8 *Let α and β have the same difference pattern Δ defined in Proposition 7, and α is not necessarily equal to β. The input (hexadecimal) difference $(0\alpha00, 0000, 0000, 0000)$ cannot propagate to the output (hexadecimal) difference $(0000, 00\beta\beta\ , 0000, 0000)$ after complete 7 rounds of Midori128.*

Proof. The difference patten of α and β implies that only the top 4-bit S-box Sb_1 of the 8-bit S-box SSb_1 will be active. From the forward direction, the difference pattern α will be preserved until the internal state x_3. Then applying SSb_0 and SSb_2 on $\Delta x_3[4]$ and $\Delta x_3[6]$, respectively, will change it in a way that guarantees that the two 4-bit S-boxes are active and applying SSb_1 on $\Delta x_3[5]$ will preserve the difference pattern α (see Proposition 7). Then,

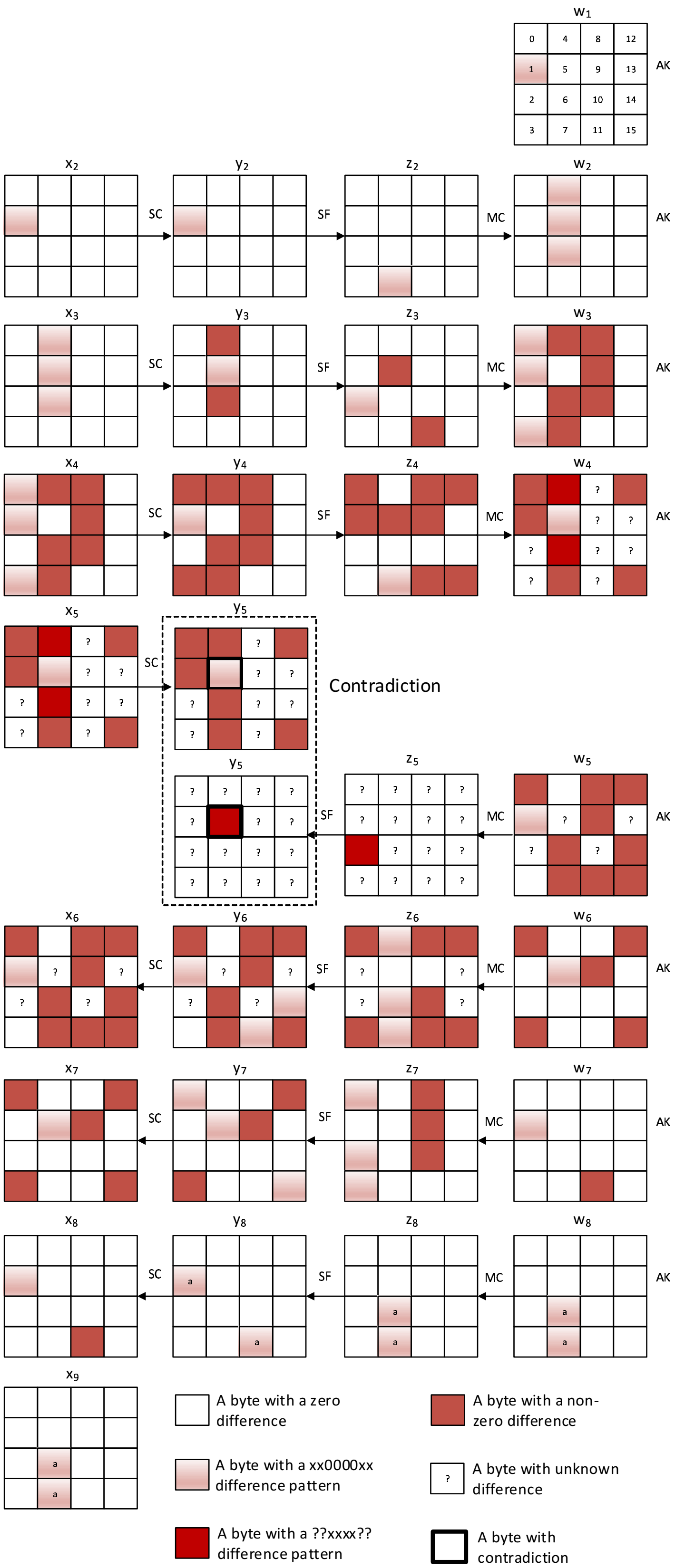

w1
0 4 8 12
1 5 9 13
2 6 10 14
3 7 11 15
AK
x2
SC
y2
SF
z2
MC
w2
AK
x3
y3
z3
w3
x4
y4
z4
w4
x5
y5
Contradiction
z5
w5
x6
y6
z6
w6
x7
y7
z7
w7
x8
y8
z8
w8
x9
?
a
A byte with a zero difference
A byte with a non-zero difference
A byte with a xx0000xx difference pattern
A byte with unknown difference
A byte with a ??xxxx?? difference pattern
A byte with contradiction

Figure 6.2: 7-round impossible differential distinguisher of Midori128

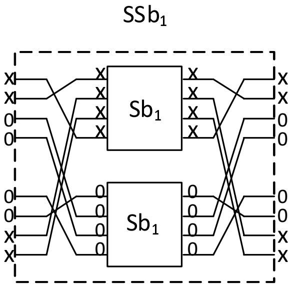


Figure 6.3: The propagation of a difference pattern $\Delta = xx0000xx$ through SSb_1

the differences in the internal state y_3 can be propagated in the same manner until state x_5 that will have only one byte preserving the α difference pattern at $\Delta y_5[5]$. From the other direction, since $\Delta x_9[6] = \Delta x_9[7]$ and each one of these differences preserve the β difference pattern, $\Delta z_8[6]$ will be equal to $\Delta z_8[7]$ and each one of these resulting differences will preserve the β difference pattern. From Proposition 7, applying SSb_1^{-1} on $\Delta y_8[1]$ preserves the β difference pattern, while applying SSb_3^{-1} on $\Delta y_8[11]$ ensures that the top and bottom 4 bits after applying the bit input permutation p_1 of SSb_1 are active. Then, the differences can be propagated in the same manner until state w_5. From the shuffle and mix column operations, we know that $\Delta y_5[5] = \Delta z_5[2] = \Delta w_5[0] \oplus \Delta w_5[1] \oplus \Delta w_5[3]$ which means that the bottom 4-bit of $p_1(\Delta y_5[5])$ are active. However, from the forward direction, we know that the bottom 4-bit of $p_1(\Delta y_5[5])$ are inactive as this byte satisfies the α difference pattern. Therefore, there is a contradiction between the byte $\Delta y_5[5]$ in the forward and backward directions, and hence the whole truncated differential characteristic holds with probability exactly 0. The previous proposition also holds for α and β have the difference pattern $00xxxx00$.

In our attack we exploit the following 4 impossible differential distinguishers:

$$(0\alpha00, 0000, 0000, 0000) \nrightarrow (0000, 00\beta\beta, 0000, 0000)(\alpha = \beta = xx0000xx) \tag{6.1}$$

$$(0\alpha00, 0000, 0000, 0000) \nrightarrow (0000, 00\beta\beta, 0000, 0000)(\alpha = \beta = 00xxxx00) \tag{6.2}$$

$$(0\alpha00, 0000, 0000, 0000) \nrightarrow (0000, \beta00\beta, 0000, 0000)(\alpha = \beta = xx0000xx) \tag{6.3}$$

$$(0\alpha00, 0000, 0000, 0000) \nrightarrow (0000, \beta00\beta, 0000, 0000)(\alpha = \beta = 00xxxx00) \tag{6.4}$$

As can be seen, all the four distinguishers above begin with one active byte which has only 4 active bits at position 1, and end with two active bytes where each byte has only 4 active bits at the same column. It should be noted that the obtained 24 distinguishers can be categorized into 6 groups, where each group contains 4 patterns similar to the presented above, and anyone of theses groups can be used in our attack instead of the above four distinguishers.

6.3.2 11-round Multiple Impossible Differential of Midori128

In this section, we present an 11-round multiple impossible differential attack against Midori128 involving both the pre-whitening and post-whitening keys, see Figure 6.4. In what follows, we describe the details of our attack which is decomposed of 2 phases: a data collection phase, where we generate enough message pairs to exclude the wrong keys involved in the analysis rounds and a key recovery phase, where we use the collected message pairs to identify the key candidates.

Data Collection. In this phase, we use the structure technique in order to reduce the data complexity of the attack. Our structure takes all the possible values in bytes $1, 2, 5, 7, 10, 11, 13, 14, 15$ while the other bytes take fixed value. Therefore, one structure generates $2^{9\times 8} \times (2^{9\times 8} - 1)/2 \approx 2^{143}$ possible message pairs. Then, we create 4 lists $L_i, 1 \leq i \leq 4$, where each list L_i contains the pairs which satisfy the plaintext and ciphertext differences of the i^{th} impossible differential. For example, L_1 is indexed by the bottom 4 bits of $p_1(P[1])$ and the following bytes of the ciphertext $C[0, 1, 2, 3, 4, 5, 6, 7, 11, 12]$. Therefore, L_1 contains $2^{143} \times 2^{-4} \times 2^{-80} = 2^{59}$ message pairs. Similarly, each one of the other lists contains 2^{59} message pairs. We take 2^m structures in order to launch the attack, and hence we have 2^{m+59} message pairs in each list L_i. Therefore, we query the encryption oracle 2^{m+72} times.

Key Recovery. We identify the key candidates by performing the following steps in parallel for each list L_i:
Step 1. Guess $\Delta z_9[12]$ and propagate it linearly forward to compute $\Delta x_{10}[13, 14, 15]$. From the knowledge of the ciphertext, compute $\Delta y_{10}[13, 14, 15]$. Using the S-box property (see Proposition 1), we can get one solution, on average, for $y_{10}[13, 14, 15]$. Therefore, we have 2^8 values for $K[13, 14, 15]$.
Step 2. For the lists L_1 and L_3 (resp. L_2 and L_4), we guess $\Delta x_1[5_t]$ (resp. $\Delta x_1[5_b]$) and propagate it linearly backward to compute $\Delta y_0[1, 11, 14]$. From the knowledge of the plaintext, we compute $\Delta x_0[1, 11, 14]$. Using the S-box property, we get one solution, on average, for $x_0[1_t, 11, 14]$ (resp. $x_0[1_b, 11, 14]$). Therefore, we have 2^4 values for $K[1_t, 11, 14]$ (resp. $K[1_b, 11, 14]$), and in total we have $2^8 \times 2^4 \times 2^{-8} = 2^4$ values for $K[1_t, 11, 13, 14, 15]$ (resp. $K[1_b, 11, 13, 14, 15]$) because we have 8-bit filter on $K[14]$.
Step 3. Guess $\Delta x_1[15]$ and propagate it linearly backward to compute $\Delta y_0[2, 7, 13]$. From the knowledge of the plaintext, compute $\Delta x_0[2, 7, 13]$. Using the S-box property, we get one solution, on average, for $x_0[2, 7, 13]$. Therefore, we have 2^8 values for $K[2, 7, 13]$, and in total we have, $2^4 \times 2^8 \times 2^{-8} = 2^4$ values for $K[1_t, 2, 7, 11, 13, 14, 15]$ (resp. $K[1_b, 2, 7, 11, 13, 14\ , 15]$) corresponding to lists L_1 and L_3 (resp. L_2 and L_4) because we have an 8-bit filter on $K[13]$.
Step 4. Repeat *Step 3*, but guess $\Delta x_1[0]$ instead of $\Delta x_1[15]$. Consequently, we have 2^8 values

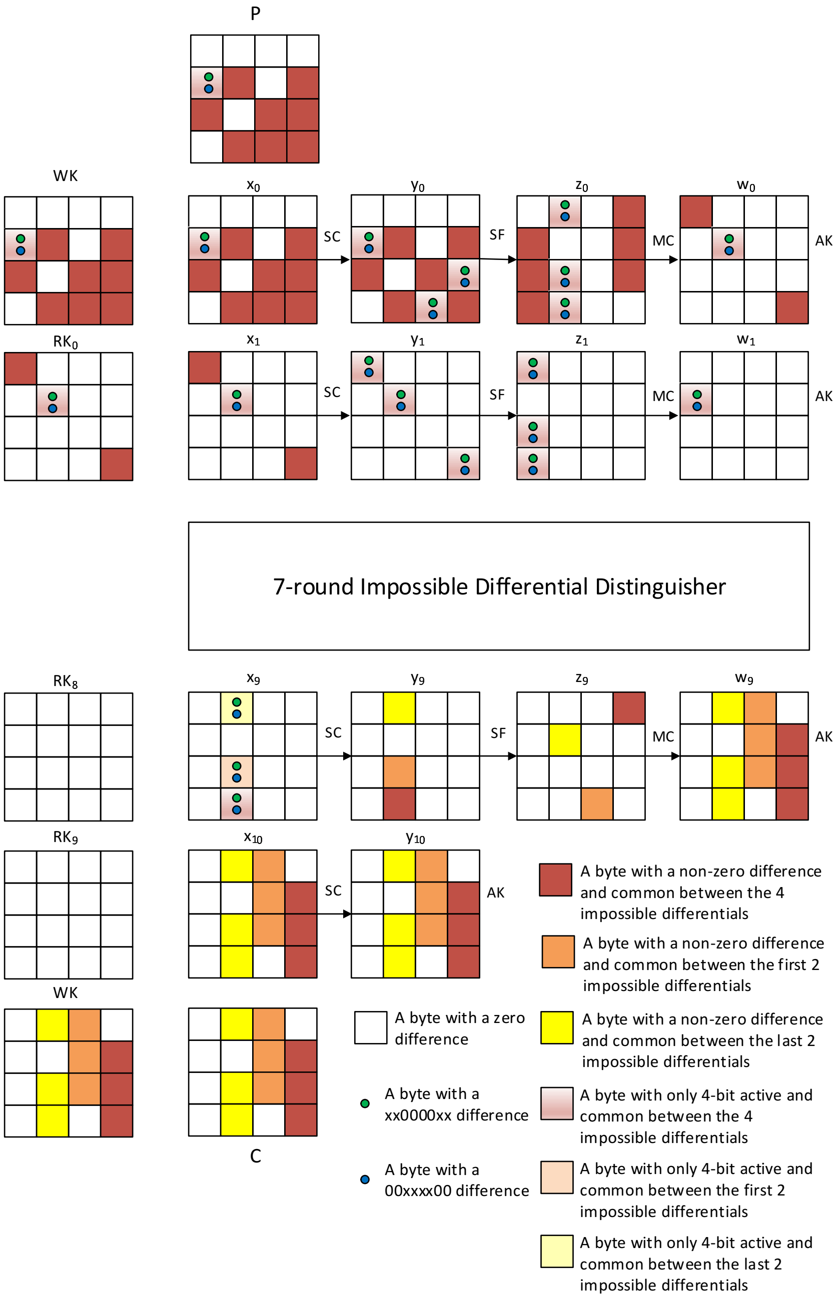

P
WK
x0
SC
y0
SF
z0
MC
w0
AK
RK0
x1
SC
y1
SF
z1
MC
w1
AK
7-round Impossible Differential Distinguisher
RK8
x9
SC
y9
SF
z9
MC
w9
AK
RK9
x10
SC
y10
AK
WK
C
A byte with a non-zero difference and common between the 4 impossible differentials
A byte with a non-zero difference and common between the first 2 impossible differentials
A byte with a zero difference
A byte with a non-zero difference and common between the last 2 impossible differentials
A byte with a xx0000xx difference
A byte with only 4-bit active and common between the 4 impossible differentials
A byte with a 00xxxx00 difference
A byte with only 4-bit active and common between the first 2 impossible differentials
A byte with only 4-bit active and common between the last 2 impossible differentials

Figure 6.4: 11-round multiple impossible differential cryptanalysis of Midori128

for $K[5,10,15]$, and in total we have $2^4 \times 2^8 \times 2^{-8} = 2^4$ values for $K[1_t, 2, 5, 7, 10, 11, 13, 14, 15]$ (resp. $K[1_b, 2, 5, 7, 10, 11\ , 13, 14, 15]$) corresponding to lists L_1 and L_3 (resp. L_2 and L_4) because we have 8-bit filter on $K[15]$.

Step 5. For the lists L_1 and L_3 (resp. L_2 and L_4), guess $\Delta w_1[1_t]$ (resp. $\Delta w_1[1_b]$) and propagate it linearly backward to deduce $K[0, 5_t, 15]$ (resp. $K[0, 5_b, 15]$) using the S-box property. Therefore, we have 2^4 values for $K[0, 5_t, 15]$ (resp. $K[0, 5_b, 15]$), and in total we have, $2^4 \times 2^4 \times 2^{-12} = 2^{-4}$ values for $K[0, 1_t, 2, 5, 7, 10, 11, 13, 14, 15]$ (resp. $K[0, 1_b, 2, 5, 7, 10, 11, 13, 14, 15]$) because we have 12-bit filter on $K[5_t, 15]$ (resp. $K[5_b, 15]$).

Step 6. For lists L_1 and L_2, guess $\Delta z_9[11]$ and propagate it linearly forward to deduce $K[8, 9, 10]$ using the S-box property. Therefore, we have 2^8 values for $K[8, 9, 10]$, and in total we have, $2^{-4} \times 2^8 \times 2^{-8} = 2^{-4}$ values for $K[0, 1_t, 2, 5, 7, 8, 9, 10, 11, 13, 14, 15]$ and $K[0, 1_b, 2, 5, 7, 8, 9, 10, 11, 13, 14, 15]$ corresponding to L_1 and L_2, respectively, because we have 8-bit filter on $K[10]$. For lists L_3 and L_4, guess $\Delta z_9[5]$ and propagate it linearly forward to deduce $K[4, 6, 7]$ using the S-box property. Therefore, we have 2^8 values for $K[4, 6, 7]$, and in total we have, $2^{-4} \times 2^8 \times 2^{-8} = 2^{-4}$ values for $K[0, 1_t, 2, 4, 5, 6, 7, 10, 11, 13, 14, 15]$ and $K[0, 1_b, 2, 4, 5\ , 6, 7, 10, 11, 13, 14, 15]$ corresponding to L_3 and L_4, respectively, because we have 8-bit filter on $K[7]$.

Step 7 For list L_1 (resp. L_2), compute $\Delta x_9[6, 7]$ and keep only the keys that make $\Delta x_9[6] = \Delta x_9[7]$ where each one has the $xx0000xx$ (resp. $00xxxx00$) difference pattern. Therefore, we have $2^{-4} \times 2^{-12} = 2^{-16}$ values for $K[0, 1_t, 2, 5, 7, 8, 9, 10, 11, 13, 14, 15]$ (resp. $K[0, 1_b, 2, 5, 7, 8, 9, 10, 11, 13, 14\ , 15]$), i.e., we remove 1 key value for the 92-bit key $K[0, 1_t, 2, 5, 7, 8, 9, 10, 11, 13\ , 14, 15]$ (resp. $K[0, 1_b\ , 2, 5, 7, 8, 9, 10, 11, 13, 14, 15]$) after processing 2^{16} message pairs. For list L_3 (resp. L_4), compute $\Delta x_9[4, 7]$ and keep only the keys that make $\Delta x_9[4] = \Delta x_9[7]$ and each one has the $xx0000xx$ (resp. $00xxxx00$) difference pattern. Therefore, we have $2^{-4} \times 2^{-12} = 2^{-16}$ values for $K[0, 1_t, 2, 4, 5, 6, 7, 10, 11, 13, 14, 15]$ (resp. $K[0, 1_b, 2, 4, 5, 6, 7, 10, 11, 13, 14, 15]$).

Attack Complexity. As explained in the previous steps, for each list L_i, we have 92 key bits involved in the analysis rounds, and for each message pair we remove, on average, 2^{-16} values. Therefore, the probability that a wrong key is not discarded for each message pair is $1 - 2^{-16}/2^{92} = 1 - 2^{-108}$. Hence, after processing all the 2^{m+59} message pairs, we have $2^{92} \times (1 - 2^{-108})^{2^{m+59}} \approx 2^{92} \times (e^{-1})^{2^{m+59-108}} \approx 2^{92} \times 2^{-1.44 \times 2^{m-49}}$ remaining candidates for 92 bits of the key. For each list L_i, in order to balance between the time and data complexities, we evaluated the time complexity of the previous steps as a function of m (see Table 6.2). As a result, we choose $m = 49$. Hence, for each list L_i, we have $2^{90.56}$ remaining key candidates for 92 bits of the key. We have 88 bits in common between L_1 and L_2, and 88 bits in common between L_3 and L_4. Therefore, we have only $2^{90.56} \times 2^{90.56} \times 2^{-88} = 2^{93.12}$ remaining key candidates for 96-bit of $K[0, 1, 2, 5, 7, 8, 9, 10, 11, 13, 14, 15]$, and $2^{93.12}$ remaining key candidates for 96 bits of $K[0, 1, 2, 4, 5, 6, 7, 10, 11, 13, 14, 15]$. For the resulting 2 lists, we have 80 bits in common. Finally, we have $2^{93.12} \times 2^{93.12} \times 2^{-80} = 2^{106.24}$ remaining key candidates for 112-bit of $K[0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15]$ that can be exhaustively searched along 2^{16} values for the 16 bits of the key that are not involved in the attack. The data complexity can be determined from the data collection phase, where we take 2^m structures. Therefore, the data complexity of the attack is $2^{m+72} = 2^{49+72} = 2^{121}$ chosen plaintexts. The time complexity of Step 1 - Step 7 is $4 \times 2^{116.69} = 2^{118.69}$. Hence, the time complexity of the attack is dominated by the data collection phase and the exhaustive search which can be estimated as $2^{121} + 2^{16} \times 2^{106.24} = 2^{122.75}$. The memory complexity of the attack is dominated by storing 2^{m+59} pairs for each list L_i to remove the wrong keys. Hence, the memory complexity is given by $4 \times 4 \times 2^{49+59} = 2^{112}$ 128-bit blocks.

Table 6.2: Time complexity of the different steps, for each list L_i, of the attack on 11-round Midori128, where NK denotes the number of keys to be excluded

Step	Time Complexity (in 11-round encryptions)	NK	$m = 49$
1	$2 \times 2^{m+59} \times 2^8 \times \dfrac{6}{2 \times 16 \times 11} \approx 2^{m+62.13}$	2^8	$2^{111.13}$
2	$2 \times 2^{m+59} \times 2^{12} \times \dfrac{5}{2 \times 16 \times 11} \approx 2^{m+65.86}$	2^4	$2^{114.86}$
3	$2 \times 2^{m+59} \times 2^{12} \times \dfrac{6}{2 \times 16 \times 11} \approx 2^{m+66.13}$	2^4	$2^{115.13}$
4	$2 \times 2^{m+59} \times 2^{12} \times \dfrac{6}{2 \times 16 \times 11} \approx 2^{m+66.13}$	2^4	$2^{115.13}$
5	$2 \times 2^{m+59} \times 2^8 \times \dfrac{5}{2 \times 16 \times 11} \approx 2^{m+61.86}$	2^{-4}	$2^{110.86}$
6	$2 \times 2^{m+59} \times 2^4 \times \dfrac{6}{2 \times 16 \times 11} \approx 2^{m+58.13}$	2^{-4}	$2^{107.13}$
7	$2 \times 2^{m+59} \times 2^{-4} \times \dfrac{4}{2 \times 16 \times 11} \approx 2^{m+49.54}$	2^{-16}	$2^{98.54}$

It should be noted that the use of four impossible differential distinguishers allows us to use the minimum possible data complexity which results in only one remaining message pair that satisfy the plaintext, ciphertext differences and the 2^{-108} probability of discarding wrong keys. This implies that we have, for each one of the corresponding lists, $L_i, 1 \leq i \leq 4$, a $2^{-1.44}$ filtration for the 92 key bits involved in the attack. Using the intersection between these four lists increases the sieving of remaining key candidates. On the other hand, increasing the number of impossible differential distinguishers to $v > 4$, with the same data complexity, will reduce the time complexity of the exhaustive search of the remaining candidates as it excludes more candidates, and increases the time complexity of Step 1 - Step 7 of the attack to $v \times 2^{116.69}$. Therefore, using $v > 4$ impossible differential distinguishers can only achieve some slight improvement in the overall time complexity of the attack.

6.4 Truncated and Multiple Differential Cryptanalysis of Reduced Round Midori128

6.4.1 A 10-round Differential of Midori128

As mentioned above, in order to minimize the number of active S-boxes, we consider differentials that have only single bit differences in the input and output of the active 8-bit S-boxes. In this section, we describe and analyze the algorithm we use to efficiently find such differentials whose probabilities are greater than 2^{-128}. In each round, we have 4 operations. The operations that can disturb the single bit difference propagation patterns are the *SubCell* and *MixColumn* operations. From the structure of the S-boxes, it follows that for any active S-box and for a given 1-bit input difference, there are at most 4 possible output differences of 1-bit difference because only 1 4-bit S-box will be active. Furthermore, from the properties of the binary almost MDS matrix, preserving single bit differences propagation patterns requires that we restrict the bit differences of the output of the active S-boxes, which lie in the same column after the shuffle operation, to be in the same position. As a result, the active bytes in each column after the *MixColumn* operation have the same value. Therefore, at each round we have at most $(8 \times 15)^4 \approx 2^{28}$ possible input differences. The term 8 in the previous formula denotes the number of possible values of the difference in each column $(1, 2, 4, \cdots, 128)$. The term 15 denotes the total number of combinations for active bytes within each column and the exponent 4 denotes the total number of columns. As noted above, for a given input difference $\mathit{\Delta} S_i$ at the beginning of each round, after the S-box layer the values of the active bytes, which lie in the same column after the shuffle operation, should be equal. Therefore, for each input difference i, we have a set Ω_i which contains at most 4^4 possible output differences (in each column we have at most four 1-bit differences, and we can have at most four active columns.)

Algorithm 8 describes the procedure used to find the maximum number of rounds r, such that a differential with the above S-box propagation patterns exists and its differential probability is greater than 2^{-128}. The algorithm run time is upper bounded by $2^{28} \times 4^4 \times r$. It utilizes four tables where each table has 2^{28} entries corresponding to the possible input differences at each round. In what follows we describe the use of each table:

1. Each entry i in the table *InputDiffProb* indicates the probability to reach the difference i at the beginning of the considered round.

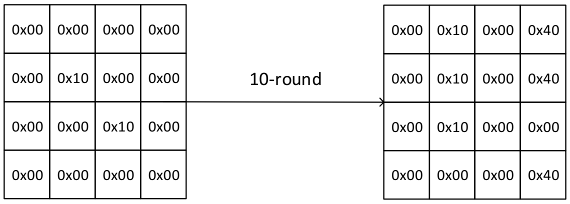
2. Each entry i in the table *OutputDiffProb* indicates the probability to reach the difference i at the end of the considered round.

3. Each entry i in the table *InputParent* indicates the input difference used at the beginning of round 0 to reach difference i at the beginning of the considered round. The value -1 is used to indicate that the difference i cannot be reached from any input difference.

4. Each entry i in the table *OutputParent* indicates the input difference used at the beginning of round 0 to reach difference i at the end of the considered round. The value -1 indicates that the difference i cannot be reached from any input difference.

As explained in Algorithm 8, we iterate over the input differences that have differential probability > 0 round by round, i.e., at each round we propagate all the input differences and begin with the obtained differences as input to the next round. In our implementation, the 28-bit index i encodes the state difference $\varDelta S$ as follows: we use 7 bits for each one of the four columns where these 7 bits are divided into two parts. The first 3 bits represent the value of the difference of the active bytes in the column and the remaining 4 bits represent the different 15 combinations of the active bytes within the column. After applying the 3 operations: *SubCell*, *ShuffleCell*, and *MixColumn*, we have 3 cases: (i) this difference did not appear before (*Outparent*[j] = -1). In this case we set its probability and parent, (ii) this difference appeared previously and its previous parent is the same as the new parent. Therefore, we add the probabilities, and (iii) this difference appears previously but with a different parent. In this case, we choose the parent with the higher probability. At the end of each round, we copy the *OutputDiffProb* and *OutParent* into *InputDiffProb* and *InputParent*, respectively, and initialize *OutputDiffProb* and *OutParent* to begin another round.

Using a PC with Intel(R) Xeon(R) CPU E3-1280 V2 @ 3.6 GHz and 32 GB RAM, a non-optimized implementation of Algorithm 8 terminates in about 3 hours and outputs $r = 10$ rounds. A 10-round differential which holds with probability 2^{-118} is illustrated in Figure 6.5 . This 10-round differential has 2554 characteristics that are distributed as shown in Table 6.3. The characteristic that holds with probability 2^{-123} is detailed in Table 6.4.



0x00	0x00	0x00	0x00
0x00	0x10	0x00	0x00
0x00	0x00	0x10	0x00
0x00	0x00	0x00	0x00

0x00	0x10	0x00	0x40
0x00	0x10	0x00	0x40
0x00	0x10	0x00	0x00
0x00	0x00	0x00	0x40

Figure 6.5: A 10 rounds differential of Midori128

Table 6.3: The characteristics distribution of the 10-round differential

# of characteristics	Probability of each characteristic
1	2^{-123}
7	2^{-124}
23	2^{-125}
50	2^{-126}
83	2^{-127}
2390	$< 2^{-127}$

Table 6.4: The 2^{-123} 10-round characteristic of Midori128

i	Input difference at round i (in hexadecimal)
0	00000000 00100000 00001000 00000000
1	00000000 00000000 00000000 00080800
2	40004040 00000000 00101010 00000000
3	40400040 40400000 08080008 08080000
4	00400040 00000040 08000000 08000800
5	40000040 00404000 08000008 00000000
6	40404000 00400040 08080008 08000800
7	00400000 40004000 00080000 08000800
8	00000000 00400000 00000800 00000000
9	00000000 00000000 00000000 00080800
10	40004040 00000000 00101010 00000000

6.4.2 13-round Truncated Differential Cryptanalysis of Midori128

In this section, we show how we can extend the differential obtained in the previous section by two rounds above, and one round below, to obtain a truncated differential over 13 rounds (see Figure 6.6). Then, we present a key recovery attack on Midori128 reduced to 13 rounds using this truncated differential.

The total probability of the 13-round differential can be calculated from its three parts. First, the differential probability of the top two rounds which is 2^{-112} and can be computed as follows: (i) $4 \rightarrow 2, 3 \rightarrow 1, 4 \rightarrow 2$, and $3 \rightarrow 1$ transitions over *MixColumn* $(z_0 \rightarrow w_0)$ which happens with probability $2^{-16} \times 2^{-16} \times 2^{-16} \times 2^{-16} = 2^{-64}$, (ii) $3 \rightarrow 1$ and $3 \rightarrow 1$ transitions over *MixColumn* $(z_1 \rightarrow w_1)$ which happens with probability $2^{-16} \times 2^{-16} = 2^{-32}$ and (iii) $w_1[5]$ and $w_1[10]$ equal difference 16 which happens with probability $2^{-8} \times 2^{-8} = 2^{-16}$. Second, the differential probability of 10-round is 2^{-118}, calculated by Algorithm 8. Third, the bottom 1 round has a differential probability equals to 1. Therefore, the differential probability of the 13-round is 2^{-230}.

In what follows, we show how we can perform our key recovery attack on Midori128 reduced to 13 rounds using the above differential. The attack is decomposed of two steps. The first step is the *data collection* in which we collect many pairs of messages to guarantee that at least one of them confirms to the 13-round truncated differential in Figure 6.6. The second step is the *key recovery* in which the collected data pairs are used to identify the key candidates.

Data Collection. To reduce the number of required chosen plaintext and get enough pairs to launch the attack, we use the structure technique. Here, our structure takes all the possible values in all the bytes except bytes 2 and 11. These bytes take fixed value. Therefore, one structure generates $2^{14\times 8} \times (2^{14\times 8} - 1)/2 \approx 2^{223}$ possible pairs. We need to collect 2^{230} message pairs because the total probability of the 13-round differential is 2^{-230}. Since each structure contains 2^{223} message pairs, we need to collect 2^{7} structures to find the right pair. Therefore, we ask the encryption oracle for the encryption of 2^{119} messages.

Key Recovery. In this step, we try to identify the key candidates that confirm to the 10-round differential. First, we try to identify the number of key suggestions of 26 bytes $WK[0, 1, 3 \cdots 10, 12 \cdots 15]$, $RK_0[1, 3, 6, 9, 11, 14]$, and $WK[4, 5, 6\ , 12,\ 13, 15]$ that correspond to each pair of messages. This can be achieved as follows: to deduce the values of the 6 bytes $WK[4, 5, 6, 12, 13, 15]$, we know that the 6 bytes $\Delta x_{12}[4, 5, 6, 12, 13, 15]$ only take one value, the same difference of the end of the 10-round differential since the key add layer does not change

Algorithm 8: Find the maximum number of rounds, r, that has a differential which holds with probability $> 2^{-128}$ considering only the single bit difference for the active S-boxes

```
Result: X: Input difference, Y : Output difference, Probability: The probability
        of the differential, r: The number of rounds covered by the differential
for $i \leftarrow 0$ to $2^{28} - 1$ do
    $InputDiffProb[i] \leftarrow 1$ ,$OutputDiffProb[i] \leftarrow 0$;
    $InputParent[i] \leftarrow i$ ,$OutputParent[i] \leftarrow -1$;
$r \leftarrow 0$, valid$\leftarrow$ true;
while valid do
    for $i \leftarrow 0$ to $2^{28} - 1$ do
        if InputDiffProb[i] $= 0$ then
            continue;
        map $i$ to state $\Delta S$ (see section 3);
        forall the entries in $\Omega_i$ do
            $\Delta S_{temp} \leftarrow nextvalue(\Omega_i)$;
            $prob \leftarrow$ Probability $(\Delta S \rightarrow \Delta S_{temp})$;
            if $prob = 0$ then
                continue;
            $\Delta S_{temp} \leftarrow ShuffleCell(\Delta S_{temp})$;
            $\Delta S_{temp} \leftarrow MixColumn(\Delta S_{temp})$;
            map $\Delta S_{temp}$ into index $j$;
            if $OutParent[j] = -1$ then
                $OutputDiffProb[j] = InputDiffProb[i] \times prob$;
                $OutputParent[j] = InputParent[i]$;
            else if $OutParent[j] = InputParent[i]$ then
                $OutputDiffProb[j] = InputDiffProb[i] \times prob + OutputDiffProb[j]$;
            else
                if InputDiffProb[i]$\times$ prob $>$ OutputDiffProb[j] then
                    $OutputDiffProb[j] = InputDiffProb[i] \times prob$;
                    $OutputParent[j] = InputParent[i]$;
    $InputDiffProb \leftarrow OutputDiffProb$, $InputParent \leftarrow OutParent$;
    initialize OutputDiffProb to 0 , initialize OutputParent to -1;
    get the index $l$ of the maximum entry in InputDiffProb;
    if InputDiffProb[l] $<= 2^{-128}$ then
        valid $\leftarrow$ false;
    $r \leftarrow r + 1$;
Get the index $l$ of the maximum entry in InputDiffProb;
$X \leftarrow InputParent[l]$ ,$Y \leftarrow l$ ,$Probability \leftarrow InputDiffProb[l]$;
```

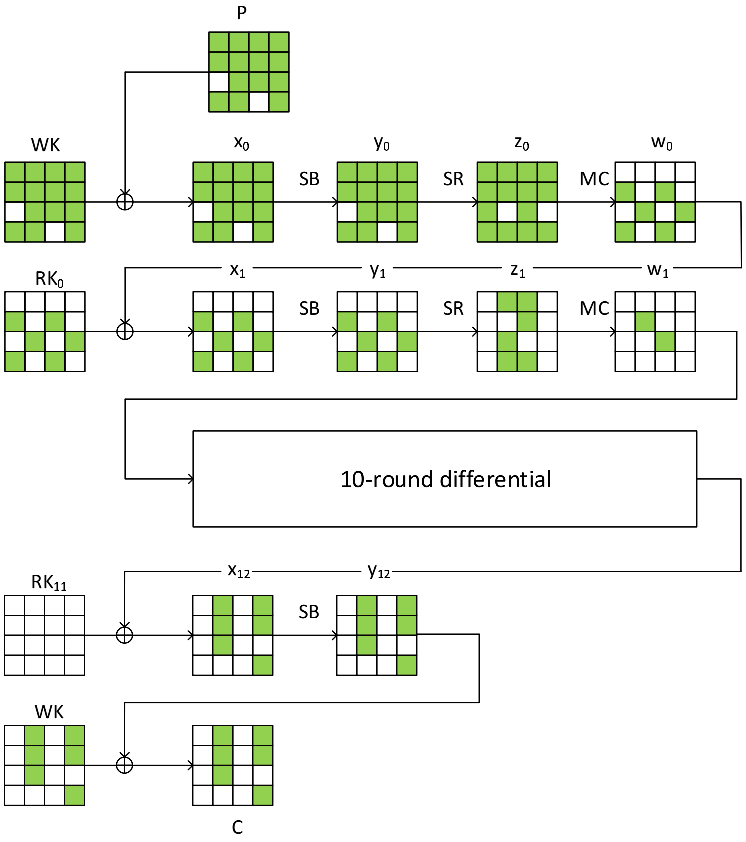


Figure 6.6: 13-round truncated differential of Midori128

the difference. The knowledge of the ciphertext allows to compute $\Delta y_{12}[4, 5, 6, 12, 13, 15]$. Using the differential property of the S-box, we can evaluate $y_{12}[4, 5, 6, 12, 13, 15]$. The knowledge of the ciphertext with $y_{12}[4, 5, 6, 12, 13, 15]$ allows us to deduce the values of the 6 bytes $WK[4, 5, 6, 12, 13, 15]$. From the other side, we know the difference ΔW_1 since it is the same difference at the beginning of the 10-round differential. Then we propagate this difference linearly trough *MixColumn* and *InvShuffleCell* to get the difference Δy_1. Δy_1 has only 6 active bytes and each, after the *SubCell* operation, has only 6 possible differences. Therefore, we have $6^6 \approx 2^{15.6}$ possible differences at Δx_1. Then after the *MixColumn* and *InvShuffleCell*, we have only $2^{15.6}$ possible differences at Δy_0. The knowledge of the plaintext allows us to compute the difference at Δx_0. Then, guessing the $2^{15.6}$ possible differences of Δy_0 and using the S-box proposition, we get the value of $x_0[0, 1, 3 \cdots 10, 12 \cdots 15]$. From the knowledge of the plaintext we can drive the value of $WK[0, 1, 3 \cdots 10, 12 \cdots 15]$. As a result we have $2^{15.6}$ key candidates for $WK[0, 1, 3 \cdots 10, 12 \cdots 15]$ and $WK[4, 5, 6, 12, 13, 15]$ but we have 6 bytes common; therefore, we have only $2^{15.6-48} = 2^{-32.4}$ key candidates. To derive the 6 bytes value $RK_0[1, 3, 6, 9, 11, 14]$, we know that we have only one difference at Δy_1. Then we get one key candidate for $RK_0[1, 3, 6, 9, 11, 14]$ but also we have 5 bytes filter between $WK[0, 1, 3 \cdots 10 , 12 \cdots 15]$ and $RK0[1, 3, 6, 9, 11, 14]$, from the key schedule as the whitening key is the master key K and the round keys $RK_i = K \oplus \beta_i$ and β_i is constant. Therefore, we have only $2^{-32.4} \times 2^{-40} = 2^{-72.4}$ key candidates for each message pair. To identify the remaining key candidates for all the message pairs, we should identify the remaining message pairs after the ciphertext filter. The ciphertext has a filter probability of $2^{-112.3}$ and is computed as follows: we have 10 bytes of zero difference which have probability of 2^{-80} and the remaining 6 bytes have probability of $2^{15.7-48} = 2^{-32.3}$ since we know that each byte of the 6 active bytes in Δx_{12} has only one difference and after the S-box layer 5 bytes out of these 6 active bytes, each, has 6 possible difference and the remaining byte has 7 possible differences. Hence, we have $6^5 \times 7 = 2^{15.7}$ possible differences at Δy_{12} which also the same differences in the 6 bytes in the ciphertext. Therefore, after the ciphertext filter, we have $2^{230} \times 2^{-112.3} = 2^{117.7}$ remaining message pairs to identify the key candidates. As a result, we have $2^{117.7} \times 2^{-72.4} = 2^{45.3}$ remaining key candidates for 15 bytes of the master key K.

In order to determine efficiently the remaining key candidates for all the remaining message pairs after the ciphertext filter, we perform the following steps:

1. From the ciphertext side, we have only one value for the active bytes of Δx_{12}. Then, using the S-box proposition, we can derive the 6 bytes $WK[4, 5, 6, 12, 13, 15]$. Therefore, we have $2^{117.7}$ key candidates for $WK[4, 5, 6, 12, 13, 15] \equiv K[4, 5, 6, 12, 13, 15]$.

2. From the plaintext side, by guessing the 6 possible differences of $\Delta w_0[14]$ and propagating them backward through the linear operations *MixColumn* and *InvShuffleCell* we can know the values of $\Delta y_0[7, 8, 13]$. Hence, we can use the S-box proposition to derive the values of $x_0[7, 8, 13]$. Then, knowing the plaintext and $x_0[7, 8, 13]$ allows to derive $WK[7, 8, 13] \equiv K[7, 8, 13]$. Using the one value of the difference $\Delta y_1[14]$, we can derive the value of $x_1[14]$ using the S-box proposition. Then, we can derive $RK_0[14] \equiv K[14]$. At this stage, we have $2^{117.7}$ key candidates and we guess 6 values to derive $K[7, 8, 13, 14]$ but we have one filter of $K[13]$ between this step and the previous step. Therefore, in total, we have $2^{117.7} \times 6 \times 2^{-8} = 2^{112.3}$ remaining key candidates for $K[4 \cdots 8, 12 \cdots 15]$.

3. By guessing the 6 possible differences of $\Delta w_0[6]$ and propagating them backward through the linear operations *MixColumn* and *InvShuffleCell*, we can determine the values of $\Delta y_0[1, 4, 14]$. Hence, we can use the S-box proposition to derive the values of $x_0[1, 4, 14]$. Then, knowing the plaintext and $x_0[1, 4, 14]$ allows us to derive $WK[1, 4, 14] \equiv K[1, 4, 14]$. Using the one value of the difference $\Delta y_1[6]$, we can derive the value of $x_1[6]$ using the S-box proposition. Then, we can derive $RK_0[6] \equiv K[6]$. At this stage we have $2^{112.3}$ key candidates and we guess 6 values to derive $K[1, 4, 6, 14]$ but we have 3 filters of $K[4, 6, 14]$ between this step and the previous step. Therefore, in total we have $2^{112.3} \times 6 \times 2^{-24} = 2^{90.9}$ remaining key candidates for $K[1, 4 \cdots 8, 12 \cdots 15]$.

4. By guessing the 6^2 possible differences of $\Delta w_0[1,3]$ and propagating them backward through the linear operations *MixColumn* and *InvShuffleCell* we can determine the values of $\Delta y_0[0,5,10,15]$. Hence, we can use the S-box proposition to derive the values of $x_0[0,5,10,15]$. Then, knowing the plaintext and $x_0[0,5,10,15]$ allows us to derive $WK[0,5,10,15] \equiv K[0,5,10,15]$. Using the one value of the difference $\Delta y_1[1,3]$, we can derive the value of $x_1[1,3]$ using the S-box proposition. Then, we can derive $RK_0[1,3] \equiv K[1,3]$. At this stage we have $2^{90.9}$ key candidates and we guess 6^2 values to derive $K[0,1,3,5,10,15]$ but we have 3 filters of $K[1,5,15]$ between this step and the previous step. Therefore, in total we have $2^{90.9} \times 6^2 \times 2^{-24} = 2^{72.1}$ remaining key candidates for $K[0,1,3\cdots 8,10,12\cdots 15]$.

5. By guessing the 6^2 possible differences of $\Delta w_0[9,11]$ and propagating them backward through the linear operations *MixColumn* and *InvShuffleCell* we can determine the values of $\Delta y_0[3,6,9,12]$. Hence, we can use the S-box proposition to derive the values of $x_0[3,6,9,12]$. Then, knowing the plaintext and $x_0[3,6,9,12]$ allows us to derive $WK[3,6,9,12] \equiv K[3,6,9,12]$. Using the one value of the difference $\Delta y_1[9,11]$, we can derive the value of $x_1[9,11]$ using the S-box proposition. Then, we can derive $RK_0[9,11] \equiv K[9,11]$. At this stage we have $2^{72.1}$ key candidates and we guess 6^2 values to derive $K[3,6,9,11,12]$ but we have one filter in this step of $K[9]$ and 3 of $K[3,6,12]$ between this step and the previous step. Therefore, in total we have $2^{72.1} \times 6^2 \times 2^{-32} = 2^{45.3}$ remaining key candidates for $K[0,1,3\cdots 15]$.

Attack Complexity. The time complexity of the key recovery phase can be derived from the previous steps as follows: step 1 needs $2 \times 2 \times 2^{117.7}/(4 \times 13) = 2^{114}$ encryptions, step 2 needs $2 \times 2^{117.7} \times 6/(4 \times 13) = 2^{115.6}$ encryptions, step 3 needs $2 \times 2^{112.3} \times 6/(4 \times 13) = 2^{110.2}$ encryptions, step 4 needs $2 \times 2^{90.9} \times 6^2/(4 \times 13) = 2^{91.4}$ encryptions, step 5 needs $2 \times 2^{72.1} \times 6^2/(4 \times 13) = 2^{72.6}$ encryptions. Therefore, the time complexity to find $2^{45.3}$ key candidates for $K[0,1,3\cdots 15]$ is $2^{114} + 2^{115.6} + 2^{110.2} + 2^{91.4} + 2^{72.6} \approx 2^{115.6}$. To retrieve the master key we make an exhaustive search for the remaining key candidates with $K[2]$ which needs $2^8 \times 2^{45.3} = 2^{53.3}$ encryptions. Therefore, the time complexity of the attack is dominated by the time needed to build the required structures which is 2^{119} encryptions. The data complexity of the attack is 2^{119} chosen plaintext.

6.4.3 Multiple Differential Cryptanalysis of Midori128

In this section, we describe a multiple differential attack that offers some time-data trade-off compared to the previous attack. Using Algorithm 8 and by enumerating all the 10-round differentials that hold with probability $> 2^{-124}$, we found 700 such differentials. In here, we show how to exploit 16 of them to launch a multiple differential attack on Midori128.

These 16 differentials are shown in Table 6.5. As shown in the table, all these differentials have the same input difference, but have different output differences that are all active in the same bytes. The first differential in Table 6.5 is the differential that is used in the attack described in the previous section. Therefore, the previous attack can be applied with multiple differentials with small modifications.

In this attack, we retrieve the same key bytes as in the previous attack. The total differential probability of the 16 differential is $2^{-114.7}$. Therefore, the total differential probability of the 13-round differential is $2^{-112} \times 2^{-114.7} = 2^{-226.7}$. Consequently, we need $2^{226.7-223} = 2^{3.7}$ structures with $2^{3.7} \times 2^{112} = 2^{115.7}$ chosen plaintext. As shown in Table 6.5, $\Delta x_{12}[4, 5, 6]$ have the following possible differences $\{4, 8, 16, 32\}$ and $\Delta x_{12}[12, 13, 15]$ have the following possible differences $\{8, 16, 32, 64\}$. Therefore, after the S-box layer, we have the following: each one of $\Delta y_{12}[4, 6, 15]$ has 19 possible differences, $\Delta y_{12}[5]$ has 15 possible differences, $\Delta y_{12}[12]$ has 17 possible differences, and $\Delta y_{12}[13]$ has 20 possible differences. As a result, we have $19^3 \times 15 \times 17 \times 20 = 2^{25.1}$ possible differences at the ciphertext. Consequently, we have $2^{226.7} \times 2^{-80} \times 2^{25.1-48} = 2^{123.8}$ remaining message pairs after the ciphertext filter. The remaining key candidates for each pair are $16 \times 2^{15.6} \times 2^{-88} = 2^{-68.4}$. Therefore, the number of remaining key candidates for all the remaining message pairs, after the ciphertext filter, is given by $2^{123.8} \times 2^{-68.4} = 2^{55.4}$ which can be exhaustively searched with the remaining key byte. We can use the same steps that are used in the previous attack to determine theses $2^{55.4}$ remaining key candidates. The time complexity of the attack is dominated by step 2. At the beginning of step 2, we have $2^{123.8} \times 16 = 2^{127.8}$ key candidates for $K[4, 5, 6, 12, 13, 15]$. Therefore, the time complexity of step 2 is $2 \times 2^{127.8} \times 6/(4 \times 13) = 2^{125.7}$ encryptions. The data complexity is $2^{115.7}$ chosen plaintext.

6.5 Conclusion

In this chapter, we have presented 7-round impossible differential distinguishers on Midori128, which, unlike the previously best known one, cover the linear transformation of the last round. Then, we exploited 4 of these distinguishers to present an 11-round attack involving the pre-whitening and post-whitening keys. This attack improves the previous best known impossible differential attack on Midori128 which covers 10 rounds without the pre-whitening key. The time, data and memory complexities of the attack are $2^{122.75}$ encryptions, 2^{121} chosen plaintexts and 2^{112} 128-bit blocks. We also showed how to exploit the structure of the S-boxes and the *MixColumn* operations of Midori128 in order to obtain long differentials that use single bit difference for the inputs and outputs of the active S-boxes. Then, we developed an algorithm that can be used to efficiently enumerate all such differentials for a given number of rounds. Using this algorithm, we obtained a 10-round differential that holds

Table 6.5: 10-round differentials of Midori128

Input difference (in hexadecimal)	Output difference (in hexadecimal)	Probability
00000000 00100000 00001000 00000000	40004040 00000000 00101010 00000000	$2^{-118.09}$
00000000 00100000 00001000 00000000	08000808 00000000 00080808 00000000	$2^{-118.12}$
00000000 00100000 00001000 00000000	08000808 00000000 00202020 00000000	$2^{-118.12}$
00000000 00100000 00001000 00000000	40004040 00000000 00080808 00000000	$2^{-118.18}$
00000000 00100000 00001000 00000000	40004040 00000000 00202020 00000000	$2^{-118.18}$
00000000 00100000 00001000 00000000	10001010 00000000 00080808 00000000	$2^{-118.36}$
00000000 00100000 00001000 00000000	10001010 00000000 00202020 00000000	$2^{-118.36}$
00000000 00100000 00001000 00000000	10001010 00000000 00101010 00000000	$2^{-118.43}$
00000000 00100000 00001000 00000000	10001010 00000000 00040404 00000000	$2^{-118.8}$
00000000 00100000 00001000 00000000	40004040 00000000 00040404 00000000	$2^{-118.8}$
00000000 00100000 00001000 00000000	20002020 00000000 00101010 00000000	$2^{-118.81}$
00000000 00100000 00001000 00000000	08000808 00000000 00040404 00000000	$2^{-119.29}$
00000000 00100000 00001000 00000000	20002020 00000000 00080808 00000000	$2^{-119.81}$
00000000 00100000 00001000 00000000	20002020 00000000 00202020 00000000	$2^{-119.81}$
00000000 00100000 00001000 00000000	08000808 00000000 00101010 00000000	$2^{-120.32}$
00000000 00100000 00001000 00000000	20002020 00000000 00040404 00000000	$2^{-120.43}$

with probability 2^{-118}. By appending 2 rounds above and one round below this 10-round differential, we obtained a 13 round truncated differential and used it to launch a key recovery attack attack on 13-round reduced Midori128. The time and data complexities of the attack are 2^{119} encryptions and 2^{119} chosen plaintext. Moreover, we presented a multiple differential attack on the 13-round reduced cipher with time and data complexities of $2^{125.7}$ encryptions and $2^{115.7}$ chosen plaintext, respectively.

Chapter 7

A MitM with Efficient Enumeration Cryptanalysis of Kuznyechik

Kuznyechik is an SPN block cipher that has been chosen recently to be standardized by the Russian federation as a new GOST cipher. The cipher employs a 256-bit key which is used to generate ten 128-bit round keys. The encryption procedure updates the 16-byte state by iterating the round function for nine rounds. In this chapter, we improve the previous 5-round Meet-in-the-Middle (MitM) attack on Kuznyechik by presenting a 6-round attack using the MitM with differential enumeration technique. Unlike previous distinguishers which utilize only the structural properties of the Maximum Distance Separable (MDS) linear transformation layer of the cipher, our 3-round distinguisher is computed based on the exact values of the coefficients of this MDS transformation. More specifically, first, we identified the MDS matrix that is utilized in this cipher. Then, we find all the relations that relate between subset of the inputs and outputs of this linear transformation. Finally, we utilized one of these relations in order to find the best distinguisher that can optimize the time complexity of the attack. Also, instead of placing the distinguisher in the middle rounds of the cipher as in the previous 5-round attack, we place it at the first 3 rounds which allows us to convert the attack from the chosen ciphertext model to the chosen plaintext model. Then, to extend the distinguisher by 3 rounds, we performed the matching between the offline and online phases around the linear transformation instead of matching on a state byte.

7.1 Introduction

Kuznyechik [117] (Grasshopper in Russian) is a substitution permutation network (SPN) block cipher which has been recently selected to be standardized by the Russian federation as a new GOST cipher [5]. The current GOST R 34.12-2015 standard defines the new cipher, Kuznyechik, in addition to the old block cipher GOST 28147-89 [1] which is now named

Magma. The cipher employs a 256-bit key which is used to generate ten 128-bit round keys. The encryption procedure updates the 16-byte state by iterating the round function for nine rounds. While the encryption process of Kuznyechik follows the SPN design architecture, its key schedule employs a Feistel network to generate the round keys.

In the single-key attack model, Kuznyechik has been analyzed in [14, 32]. Furthermore, Biryukov, *et al.* [33] revealed a hidden structure in the S-box used in the cipher. A fault analysis attack on the cipher was also presented in [13].

In this chapter, we focus on improving the previous MitM attack on the cipher. More precisely, we present 6-round MitM attack on Kuznyechik[1]. This attack utilizes a 3-round truncated differential distinguisher that is based on the efficient enumeration and multiset techniques. In particular, in this distinguisher, the match between the offline and online computations is accomplished using a linear relation that relates 5 input bytes and 12 output bytes of the linear transformation layer. In the offline phase, we compute the left hand side of the relation and store the obtained results in a table. Then, in the online phase, we work out the right hand side of the relation and compare it with the stored results in order to filter out the wrong keys. This idea, which is inspired by the work in [57], enables us to extend the 3-round distinguisher by 3 rounds to launch our 6-round MitM attack. Our attack has a time complexity of 2^{231} encryptions, a memory complexity of 2^{218} 128-bit, and a data complexity of 2^{113} chosen plaintexts.

The rest of the chapter is organized as follows. Section 7.2 provides the notations used throughout the chapter and a brief description of Kuznyechik. In section 7.3, we present our attack on 6 rounds of Kuznyechik. Finally, the chapter is concluded in section 7.4.

7.2 Specifications of Kuznyechik

The following notation is used throughout the rest of the chapter:

- K: The master key.
- K_i: The 128-bit round key used in the i^{th} round.
- x_i, y_i, z_i: The 16-byte state before the substitution, linear, and key addition operations, respectively, at round i.
- $x_i[j]$: The j^{th} byte of the state x_i, $j = 0, 1, \cdots, 15$.

[1] In appendix E, we also show how the attack presented in [14] can be tweaked to work under the chosen plaintext model with less time complexity.

- $x_i[j:l]$: The bytes from j to l of x_i, $j < l$.

- Δx_i, $\Delta x_i[j]$: The difference at state x_i, and byte $x_i[j]$, respectively.

- $\cdot, +$: The multiplication and addition in $GF(2^8)$ using the irreducible polynomial $p(x) = x^8 + x^7 + x^6 + x + 1$.

- $||$: The concatenation operation.

Kuznyechik [117, 5] employs the SPN design approach. It has a block length of 128-bit and a secret key of 256 bits. The 128-bit secret key is used to generate 10 128-bit round keys through the key schedule algorithm. The internal state is updated using the round keys by applying the round function 9 times, as illustrated in Figure 7.1. The round function has the following operations [14]:

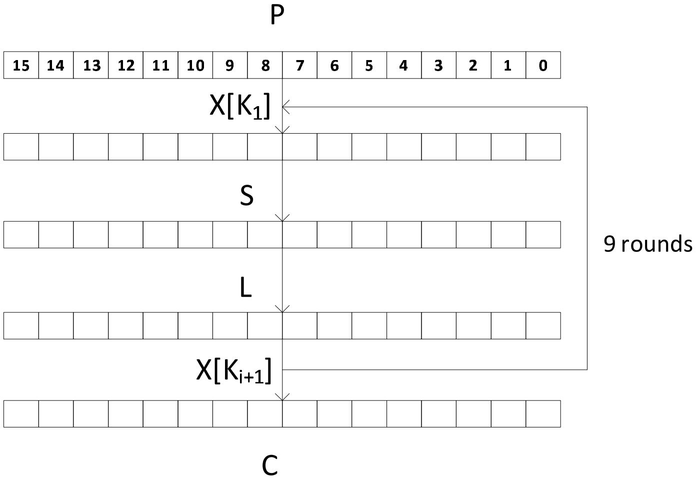


Figure 7.1: Encryption scheme

- Nonlinear bijective transformation (S): A nonlinear byte bijective mapping.

- Linear Transformation (L): An optimal diffusion operation that operates on a 16-byte input and has an optimal branch number = 17. This operation can be defined as $L(a) = R^{16}(a)$, i.e., by iterating the transformation R for 16 times on $a \in \{0,1\}^{128}$ where

$$R(a) = R(a_{15}||\cdots||a_0) = l(a_{15}||\cdots||a_0)||a_{15}||\cdots||a_1, \tag{7.1}$$

$$\begin{aligned} l(a_{15}, \cdots, a_0) = 0x94 \cdot a_{15} + 0x20 \cdot a_{14} + 0x85 \cdot a_{13} + 0x10 \cdot a_{12} \\ + 0xC2 \cdot a_{11} + 0xC0 \cdot a_{10} + a_9 + 0xFB \cdot a_8 + a_7 \\ + 0xC0 \cdot a_6 + 0xC2 \cdot a_5 + 0x10 \cdot a_4 + 0x85 \cdot a_3 \\ + 0x20 \cdot a_2 + 0x94 \cdot a_1 + a_0. \end{aligned} \tag{7.2}$$

- XOR layer (X): Mixes round keys with the encryption state.

Additionally, an XOR layer is added after the last round. Therefore, the encryption algorithm that is used to encrypt the plaintext P to obtain the ciphertext C, can be expressed as follows:

$$C = X[K_{10}] \circ (L \circ S \circ X[K_9]) \circ \cdots \circ (L \circ S \circ X[K_1])(P).$$

In our attack, we swap the order of the linear operations L, X, and hence we use an equivalent key EK_r instead of K_r such that $EK_r = L^{-1}(K_r)$.

Key Schedule. The 256-bit secret key is used to generate 10 sub-round keys of length 128 bits each through a 32-round Feistel network. First the master key K is split into two keys K_1, K_2 such that $K = K_1||K_2$. These keys are used as the first two round keys. Then, these two round keys are updated through the Feistel network that applies the same round function of the encryption on the right branch of the Feistel structure. Finally, as depicted in Figure 7.2, each pair of subsequent round keys is extracted after eight rounds of execution. The XOR operation in the round function is done using constants C_i, $i = 1, 2, \cdots, 32$, defined as $C_i = L(i)$.

We measure the memory complexity of our attacks as 128-bit Kuznyechik blocks and the time complexity in terms of the equivalent number of reduced-round Kuznyechik encryptions.

7.3 A MitM attack on 6-round Kuznyechik

Here, in order to identify the key candidates, we use the multisets (see Definition 7) of the outputs corresponding to the δ-set (see Definition 5) as a distinguishing property. Our attack depends on the S-box proposition (see Proposition 1). The distinguishing property used in our attack is based on a relation between 5 input bytes and 12 output bytes of the linear transformation L, which can be captured by the following proposition:

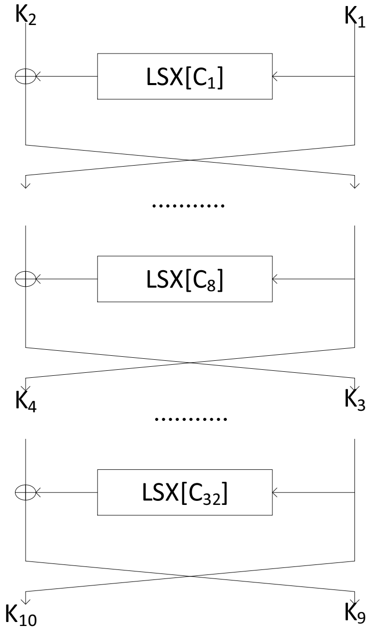


Figure 7.2: Key schedule

Proposition 9 *The linear transformation L that transform the input $a = a_{15}||a_{14}|| \cdots ||a_0$ to the output $b = b_{15}||b_{14}|| \cdots ||b_0$ has the following property:*

$$\begin{gathered} a_{11} + 0x94 \cdot a_{12} + 0x20 \cdot a_{13} + 0x85 \cdot a_{14} + 0x10 \cdot a_{15} = \\ 0xC2 \cdot b_0 + 0xC0 \cdot b_1 + b_2 + 0xFB \cdot b_3 + b_4 + 0xC0 \cdot b_5 + \\ 0xC2 \cdot b_6 + 0x10 \cdot b_7 + 0x85 \cdot b_8 + 0x20 \cdot b_9 + 0x94 \cdot b_{10} + b_{11}. \end{gathered} \tag{7.3}$$

Proposition 9 is obtained by first using equations (7.1) and (7.2) to calculate the equivalent 16×16 MDS matrix, A, that transforms the input a to the output b. Thus we have[2]

[2] All the matrix coefficients $\in GF(2^8)$ and are expressed in hexadecimal notation.

$$
\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \\ b_9 \\ b_{10} \\ b_{11} \\ b_{12} \\ b_{13} \\ b_{14} \\ b_{15} \end{pmatrix}
=
\left(\begin{array}{cccccccccccccccc}
1 & 94 & 20 & 85 & 10 & C2 & C0 & 1 & FB & 1 & C0 & C2 & 10 & 85 & 20 & 94 \\
94 & A5 & 3C & 44 & D1 & 8D & B4 & 54 & DE & 6F & 77 & 5D & 96 & 74 & 2D & 84 \\
84 & 64 & 48 & DF & D3 & 31 & A6 & 30 & E0 & 5A & 44 & 97 & CA & 75 & 99 & DD \\
DD & 0D & F8 & 52 & 91 & 64 & FF & 7B & AF & 3D & 94 & F3 & D9 & D0 & E9 & 10 \\
10 & 89 & 48 & 7F & 91 & EC & 39 & EF & 10 & BF & 60 & E9 & 30 & 5E & 95 & BD \\
BD & A2 & 48 & C6 & FE & EB & 2F & 84 & C9 & AD & 7C & 1A & 68 & BE & 9F & 27 \\
27 & 7F & C8 & 98 & F3 & 0F & 54 & 8 & F6 & EE & 12 & 8D & 2F & B8 & D4 & 5D \\
5D & 4B & 8E & 60 & 1 & 2A & 6C & 9 & 49 & AB & 8D & CB & 14 & 87 & 49 & B8 \\
B8 & 6E & 2A & D4 & B1 & 37 & AF & D4 & BE & F1 & 2E & BB & 1A & 4E & E6 & 7A \\
7A & 16 & F5 & 52 & 78 & 99 & EB & D5 & E7 & C4 & 2D & 6 & 17 & 62 & D5 & 48 \\
48 & C3 & 2 & 0E & 58 & 90 & E1 & A3 & 6E & AF & BC & C5 & 0C & EC & 76 & 6C \\
6C & 4C & DD & 65 & 1 & C4 & D4 & 8D & A4 & 2 & EB & 20 & CA & 6B & F2 & 72 \\
72 & E8 & 14 & 7 & 49 & F6 & D7 & A6 & 6A & D6 & 11 & 1C & 0C & 10 & 33 & 76 \\
76 & E3 & 30 & 9F & 6B & 30 & 63 & A1 & 2B & 1C & 43 & 68 & 70 & 87 & C8 & A2 \\
A2 & D0 & 44 & 86 & 2D & B8 & 64 & C1 & 9C & 89 & 48 & 90 & DA & C6 & 20 & 6E \\
6E & 4D & 8E & EA & A9 & F6 & BF & 0A & F3 & F2 & 8E & 93 & BF & 74 & 98 & CF
\end{array}\right)
\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \\ a_9 \\ a_{10} \\ a_{11} \\ a_{12} \\ a_{13} \\ a_{14} \\ a_{15} \end{pmatrix}
$$

Then by performing Gauss elimination on the augmented matrix $(A||b)$, we obtain

$$
\left(\begin{array}{cccccccccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
94 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
20 & 94 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
85 & 20 & 94 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
10 & 85 & 20 & 94 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
C2 & 10 & 85 & 20 & 94 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
C0 & C2 & 10 & 85 & 20 & 94 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & C0 & C2 & 10 & 85 & 20 & 94 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
FB & 1 & C0 & C2 & 10 & 85 & 20 & 94 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & FB & 1 & C0 & C2 & 10 & 85 & 20 & 94 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
C0 & 1 & FB & 1 & C0 & C2 & 10 & 85 & 20 & 94 & 1 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{C2} & \mathbf{C0} & \mathbf{1} & \mathbf{FB} & \mathbf{1} & \mathbf{C0} & \mathbf{C2} & \mathbf{10} & \mathbf{85} & \mathbf{20} & \mathbf{94} & \mathbf{1} & 0 & 0 & 0 & 0 \\
10 & C2 & C0 & 1 & FB & 1 & C0 & C2 & 10 & 85 & 20 & 94 & 1 & 0 & 0 & 0 \\
85 & 10 & C2 & C0 & 1 & FB & 1 & C0 & C2 & 10 & 85 & 20 & 94 & 1 & 0 & 0 \\
20 & 85 & 10 & C2 & C0 & 1 & FB & 1 & C0 & C2 & 10 & 85 & 20 & 94 & 1 & 0 \\
94 & 20 & 85 & 10 & C2 & C0 & 1 & FB & 1 & C0 & C2 & 10 & 85 & 20 & 94 & 1
\end{array}\right)
\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \\ b_9 \\ b_{10} \\ b_{11} \\ b_{12} \\ b_{13} \\ b_{14} \\ b_{15} \end{pmatrix}
=
\left(\begin{array}{cccccccccccccccc}
1 & 94 & 20 & 85 & 10 & C2 & C0 & 1 & FB & 1 & C0 & C2 & 10 & 85 & 20 & 94 \\
0 & 1 & 94 & 20 & 85 & 10 & C2 & C0 & 1 & FB & 1 & C0 & C2 & 10 & 85 & 20 \\
0 & 0 & 1 & 94 & 20 & 85 & 10 & C2 & C0 & 1 & FB & 1 & C0 & C2 & 10 & 85 \\
0 & 0 & 0 & 1 & 94 & 20 & 85 & 10 & C2 & C0 & 1 & FB & 1 & C0 & C2 & 10 \\
0 & 0 & 0 & 0 & 1 & 94 & 20 & 85 & 10 & C2 & C0 & 1 & FB & 1 & C0 & C2 \\
0 & 0 & 0 & 0 & 0 & 1 & 94 & 20 & 85 & 10 & C2 & C0 & 1 & FB & 1 & C0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 94 & 20 & 85 & 10 & C2 & C0 & 1 & FB & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 94 & 20 & 85 & 10 & C2 & C0 & 1 & FB \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 94 & 20 & 85 & 10 & C2 & C0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 94 & 20 & 85 & 10 & C2 & C0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 94 & 20 & 85 & 10 & C2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{94} & \mathbf{20} & \mathbf{85} & \mathbf{10} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 94 & 20 & 85 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 94 & 20 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 94 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right)
\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \\ a_9 \\ a_{10} \\ a_{11} \\ a_{12} \\ a_{13} \\ a_{14} \\ a_{15} \end{pmatrix}
\tag{7.4}
$$

It should be noted that this Gauss elimination step allows us to obtain many linear relations that relate subsets of the input and output bytes of the linear transformation, however, the relation identified by Proposition 9, which corresponds to the bold line in equation (7.4) above, helped us to reduce the total time complexity of the attack.

As depicted in Figure 7.3, the distinguisher starts at x_1 and ends at y_4. The δ-set is chosen at byte 15 and the multiset is computed from the left hand side of equation (7.3) using bytes $11, 12, 13, 14, 15$. Our distinguisher is based on the following proposition:

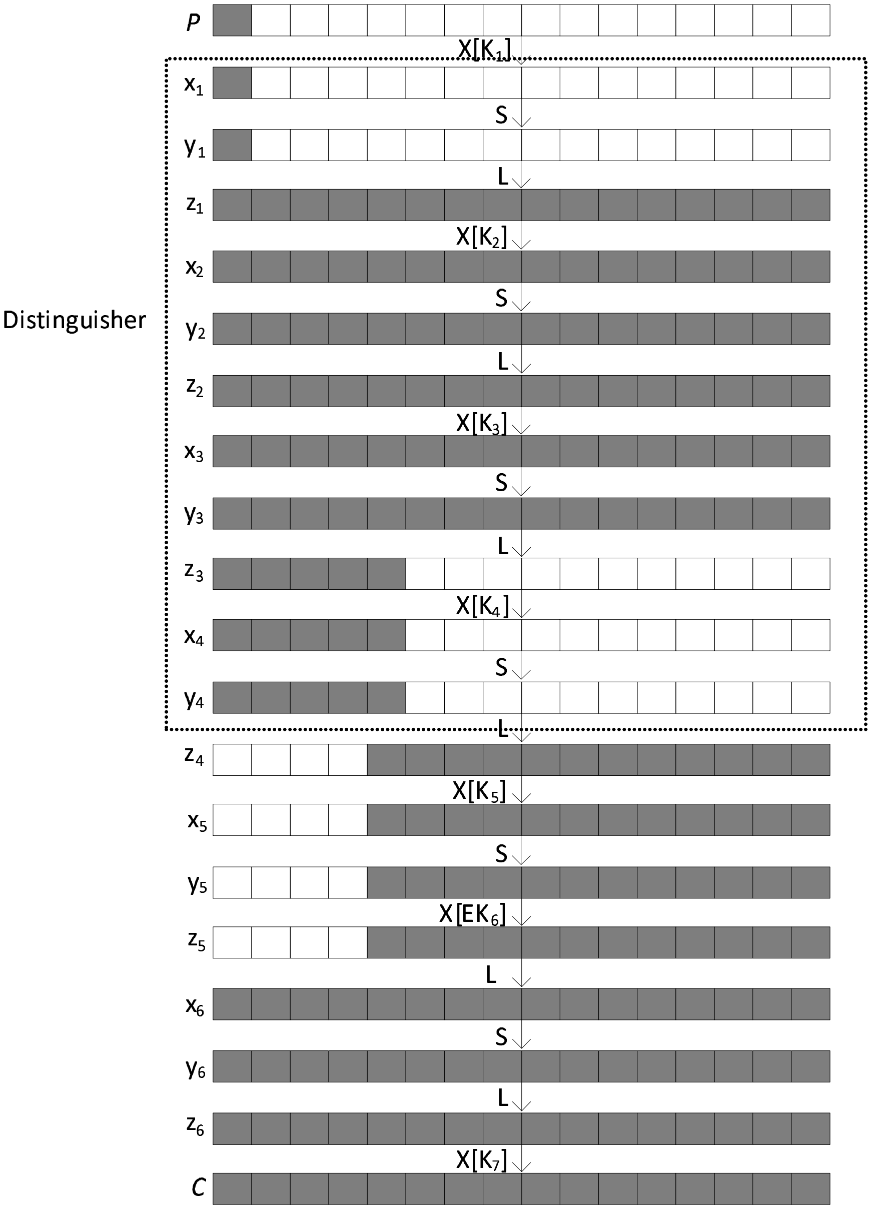


Figure 7.3: Kuznyechik 6-round attack

Proposition 10 *If a message m belongs to a pair of states conforming to the truncated* *differential characteristic shown in Figure 7.3, then the multiset of differences* $y_4[11] + 0x94 \cdot y_4[12] + 0x20 \cdot y_4[13] + 0x85 \cdot y_4[14] + 0x10 \cdot y_4[15]$ *obtained from the* δ*-set constructed from* m *in* $x_1[15]$ *is fully determined by the following 27 bytes:* $\Delta x_1[15], x_2, y_4[11:15]$ *and* $\Delta y_4[11:15]$.

Proof. The proof is based on the efficient enumeration technique [58]. In the following, we show how the knowledge of 27 bytes is enough to propagate the δ-set at x_1 and compute the multiset $(y_4[11] + 0x94 \cdot y_4[12] + 0x20 \cdot y_4[13] + 0x85 \cdot y_4[14] + 0x10 \cdot y_4[15])$. Let (m, m') be a right pair that conforms to the truncated differential characteristic in Figure 7.3. Since we are interested in the multiset instead of the ordered sequence and the S-box that is used is bijective, then we can propagate the difference $\Delta y_1[15]$ instead of $\Delta x_1[15]$. $\Delta y_1[15]$ can be propagated through the linear operations L, and X to compute Δx_2. With the knowledge of x_2, we can bypass the non-linear operation S. Then we can forward the knowledge through L, X to get Δx_3. Similarly, in the backward direction, the knowledge of $y_4[11:15]$ allows the propagation of $\Delta y_4[11:15]$ through the non-linear mapping S^{-1} to get $\Delta x_4[11:15]$. Then, we propagate the difference $\Delta x_4[11:15]$ backward linearly through X, L^{-1} to determine Δy_3. Using the differential property of Kuznyechik S-box, the expected number of solutions for x_3 is one.

The previous 3-round distinguisher is utilized to launch a 6-round attack on Kuznyechik by appending 3 rounds below it. In what follows, we describe the details of our attack which has two phases, namely the offline and online phases.

Offline Phase. In this phase, we build the distinguishing property that will be used in the online phase to filter the key space. As mentioned above, our distinguisher can be built using only 27 parameters given by Proposition 10. First, we iterate on the $2^{27 \times 8} = 2^{216}$ possible values for each of the 27 byte parameters, to deduce the internal state variable x_3. Then, using the obtained value of x_3, we propagate the δ-set to compute the multiset differences $y_4[11] + 0x94 \cdot y_4[12] + 0x20 \cdot y_4[13] + 0x85 \cdot y_4[14] + 0x10 \cdot y_4[15]$ and store them in a table. Consequently, we have 2^{216} multisets out of the $2^{467.6}$ theoretically possible ones (not $2^{506.17}$ because the number of ordered sequences associated to a multiset is not constant [58]).

Online Phase. In this phase, we have two steps. First, the *data collection* step where we want to ensure that we have one pair of messages that conform to the truncated differential characteristic in Figure 7.3, which holds with probability p. This is achieved by collecting $1/p$ pair of messages. Second, the *key recovery* where we first identify the key values that satisfy the truncated differential characteristic in Figure 7.3. Then these key values with the collected data pairs are used to create the δ-set, compute the multiset in the online phase and compare it with the precomputation table to determine the valid key candidates.

Data Collection. We employed our δ-set in the plaintext side to operate in the chosen plaintext model. We also utilize plaintext structures to reduce the amount of required plaintexts. The utilized structure takes all the possible values in byte 15 while the remaining bytes take fixed value. Therefore, one structure generates $2^8 \times (2^8 - 1)/2 \approx 2^{15}$ possible pairs. The probability of the whole truncated differential characteristic of Figure 7.3 can be calculated from the following probabilities: transition from x_6 to z_5 over L^{-1} $(16 \rightarrow 12)$ of probability $2^{-4\times 8} = 2^{-32}$ and transition from z_4 to y_4 over L^{-1} $(12 \rightarrow 5)$ of probability $2^{-11\times 8} = 2^{-88}$. Therefore, the total probability of the truncated differential characteristic is 2^{-120}. Hence, to find one pair of messages that conform to the truncated differential characteristic, we need to collect 2^{120} message pairs. Since each structure contains 2^{15} message pairs, 2^{105} structures will suffice to find the right pair. Therefore, the total number of queries to the encryption oracle is about 2^{113}.

Key Recovery. This step involves two sub steps. First, identifying the key suggestions that will be used in building and computing the δ-set and the multiset, respectively. Second, checking whether the identified keys in the first sub step are key candidates. The keys that are used in building the δ-set and computing the multiset are $EK_6[0:11], K_7$, i.e., we want to guess 28 bytes which will make the computational complexity exceeds the exhaustive search. Instead, we identify the number of key suggestions of the 28 bytes that correspond to each pair of messages. This can be achieved as follows: to deduce the values of the 16 bytes of K_7, we guess 12 bytes of $\Delta y_5[0:11]$ and propagate these values linearly through X, L to get Δx_6. The knowledge of the ciphertext allows us to compute Δy_6. Using the differential property of the S-box, we evaluate y_6. Then, we compute z_6 from y_6 through the linear operation L. The knowledge of the ciphertext and z_6 allows us to deduce the values of the 16 bytes of K_7. For the next 12 bytes of $Ek_6[0:11]$, the 5 bytes of Δy_4, $\Delta y_4[11:15]$ can take 2^{40} values, but only 2^8 values among them can follow the truncated differential characteristic and after applying L, Δz_4 has only 4 zero bytes. Therefore, to deduce the values of the 12 bytes $EK_6[0:11]$, we guess the 2^8 values for $\Delta y_4[11:15]$ and propagate them linearly forward through L, X to compute $\Delta x_5[0:11]$. The knowledge of $\Delta x_5[0:11]$ and $\Delta y_5[0:11]$ allows us to deduce the value of 12 bytes $y_5[0:11]$ using the differential property of the S-box. We can also propagate y_6 through S^{-1} and L^{-1} to get $z_5[0:11]$. The knowledge of $z_5[0:11]$ and $y_5[0:11]$ allows to deduce $EK_6[0:11]$. To summarize this part, we have $2^{13\times 8} = 2^{104}$ key suggestions for the 28 key bytes $EK_6[0:11], K_7$.

Now, we can use the 2^{120} collected message pairs and the 2^{104} suggestions for the 28 key bytes $EK_6[0:11], K_7$ to identify the δ-set and compute the multiset constructed from $0xC2 \cdot z_4[0] + 0xC0 \cdot z_4[1] + z_4[2] + 0xFB \cdot z_4[3] + z_4[4] + 0xC0 \cdot z_4[5] + 0xC2 \cdot z_4[6] + 0x10 \cdot$

$z_4[7] + 0x85 \cdot z_4[8] + 0x20 \cdot z_4[9] + 0x94 \cdot z_4[10] + z_4[11]$. Consequently, the key suggestion is considered wrong if no match is found in the table. Otherwise, it is considered as a key candidate. The probability of a wrong key leading to a match in the table is $2^{216}/2^{467.6} = 2^{-251.6}$. Therefore, after this step we will have $2^{120+104-251.6} = 2^{-27.6}$ key candidates for the key bytes $EK_6[0:11], K_7$. In other words, only the right candidate will pass this filtering step. From the key schedule of Kuznyechik, the master key can be recovered by guessing K_8. Then, we can propagate the knowledge of K_7 and K_8 until we evaluate K_1 and K_2, and hence the master key $K = K_1||K_2$. Therefore, retrieving the master key requires to exhaustively search the remaining key candidate of K_7 with the 2^{128} possible values of K_8 using two plaintext/ciphertext pairs.

Attack Complexity. The offline phase dominates the memory complexity of the attack in which we store 2^{216} multiset where each multiset is 512 bits. Therefore, the memory complexity of the attack is $2^{216} \times 512/128 = 2^{218}$ 128-bit blocks. The data collection step which determines the data complexity of the attack requires us to collect 2^{113} plaintext to generate 2^{120} message pairs. Hence the data complexity of the attack is 2^{113} chosen plaintexts. The time complexity of the attack has three main components. First, the time needed to store the precomputation table, is $2^{27\times 8} \times 2^8 \times 3/6 = 2^{223}$ encryptions. Second, the time required to get the one key candidate for K_7 is $2^{120} \times 2^{13\times 8} \times 2^8 \times 3/6 = 2^{231}$ encryptions. Third, the time required to retrieve the master key is $2 \times 2^{128} = 2^{129}$. Therefore, the time complexity of the attack is about 2^{231} encryptions.

7.4 Conclusion

In this chapter, we presented a Meet-in-the-Middle attack on 6-round reduced Kuznyechik. By exploiting the exact values of the coefficients of the MDS transformation of the cipher, our attack recovers the master key with data complexity of 2^{113} chosen plaintexts, time complexity of 2^{231} encryptions, and memory complexity of 2^{218} 128-bit blocks. This attack improves the previous results of the MitM with differential enumeration technique which can only reach 5 rounds. Compared to the best known attack which does not require the full code book on this cipher in the single-key model, and which also reaches 6 rounds, our attack improves the data complexity at the expense of requiring more time and memory complexities.

Chapter 8

Multidimensional Zero-Correlation Attacks on SPARX-128

SPARX is a family of ARX-based block ciphers proposed at ASIACRYPT 2016. This family was designed with the aim of providing provable security against single-characteristic linear and differential cryptanalysis. SPARX-128/128 and SPARX-128/256 are two members of this family which operate on data blocks of length 128 bits and keys of length 128 and 256 bits, respectively. In this work, we propose a zero-correlation distinguisher that covers 5 steps (20 rounds) for both variants of SPARX-128. Then, using specific linear masks at its output and utilizing some properties of the employed linear layer and S-box, we extend this distinguisher to 5.25 steps (21 rounds). By exploiting some properties of the key schedule, we extend the 20-round distinguisher by 4 rounds to present a 24-round multidimensional zero-correlation attack against SPARX-128/256, i.e., 6 steps out of 10 steps. The 24-round attack is then extended to a 25-round (6.25 out of 10 steps) zero-correlation attack against SPARX-128/256 with the full codebook by using the developed 21-round distinguisher. In addition, we extend the 21-round distinguisher by one round to launch a 22-round multidimensional zero-correlation attack against SPARX-128/128, i.e., 5.5 steps out of 8 steps.

8.1 Introduction

With the aim of developing block ciphers with provable security against single-characteristic linear and differential cryptanalysis, Dinu *et al*. [61] proposed a new ARX-based family of block ciphers at ASIACRYPT 2016. They achieved this goal by proposing a new strategy, namely, the long trail strategy, which is different from the well-studied wide trail strategy [53] that is used by many S-box based block ciphers. The long trail strategy encourages the use of a rather weak but large S-boxes such as ARX-based S-boxes along with a very light linear transformation layer. Adopting this strategy in the SPARX family allowed the designers to prove the security of the cipher against single-characteristic linear and differential cryptanalysis by bounding the maximum linear and differential probabilities for any number of rounds.

SPARX-128/128 and SPARX-128/256 are two members of the SPARX family which employ a data block of length 128 bits using 128 and 256 key bits, respectively. The only known attacks against these two variants were developed by the designers. These attacks were found using integral cryptanalysis based on Todo's division property [125] and cover 22 and 24 rounds of SPARX-128/128 and SPARX-128/256, respectively, in the chosen plaintext attack model.

Zero-correlation [39] is one of the relatively new techniques that is used to analyze symmetric-key primitives, where the attacker utilizes a linear approximation of probability exactly 1/2 over r_m rounds to act as a distinguisher. Then, this distinguisher can be utilized in a key recovery attack such that the keys which lead to this distinguisher are excluded. This technique proves its success against many of the recently proposed block ciphers as exemplified by the work done in [138, 120, 39, 141, 135].

In this chapter, we evaluate the security of SPARX-128 in the known plaintext attack model using the zero-correlation cryptanalysis. First, we present a 20-round zero-correlation distinguisher. Then, we use a specific linear mask at the output of this 20-round distinguisher and exploit some properties of the employed linear layer and S-box to add one more round and create a 21-round zero-correlation distinguisher. To turn these distinguishers into key recovery attacks, we take advantage of the property of the S-box that permits the existence of a two-round linear approximation that holds with probability 1. Then, by exploiting the key schedule relations, we place this deterministic two-round linear approximation in a position that enables us to extend the 20-round distinguisher by 4 complete rounds, i.e., including the linear layer, to launch a 24-round key recovery attack against SPARX-128/256 using multidimensional zero-correlation attack. This 24-round attack is, then, extended by one more round using the 21-round distinguisher to launch a 25-round zero-correlation attack against SPARX-128/256 using the full codebook. In addition, we extend the 21-round distinguisher to launch a 22-round attack against SPARX-128/128.

The remainder of the chapter is organized as follows. In section 8.2, the notations used throughout the chapter and the specifications of SPARX-128/128 and SPARX-128/256 are presented. In section 8.3, we present our distinguisher for SPARX-128/128 and SPARX-128/256. Afterwards, in section 8.4, we provide a detailed description of our multidimensional zero-correlation attacks against SPARX-128/128 and SPARX-128/256, and finally we conclude the chapter in section 8.5.

8.2 Specifications of SPARX-128

The following notations are used throughout the chapter:

- K: The master key.

- k_i: The i^{th} 16-bit of the key state, where $0 \leq i \leq 7$ for SPARX-128/128, and $0 \leq i \leq 15$ for SPARX-128/256.

- K_i: The i^{th} 32-bit of the key state, where $0 \leq i \leq 3$ for SPARX-128/128, and $0 \leq i \leq 7$ for SPARX-128/256.

- k_i^j: The i^{th} 16-bit of the key state after applying the key schedule permutation j times, where $0 \leq i \leq 7$, $0 \leq j \leq 32$ for SPARX-128/128, and $0 \leq i \leq 15$, $0 \leq j \leq 20$ for SPARX-128/256.

- K_i^j: The i^{th} 32-bit of the key state after applying the key schedule permutation j times, where $0 \leq i \leq 3$, $0 \leq j \leq 32$ for SPARX-128/128, and $0 \leq i \leq 7$, $0 \leq j \leq 20$ for SPARX-128/256.

- $RK_{(a,i)}$: The 32-bit round key used at branch a of round i where $0 \leq i \leq 32$ (resp. $0 \leq i \leq 40$) for SPARX-128/128 (resp. SPARX-128/256), and $0 \leq a \leq 3$, with $a = 0$ corresponding to the left branch.

- $X_{(a,i)}$ ($Y_{(a,i)}$): The left (right) 16-bit input at branch a of round i where $0 \leq i \leq 32$ (resp. $0 \leq i \leq 40$) for SPARX-128/128 (resp. SPARX-128/256), $0 \leq a \leq 3$, with $a = 0$ corresponding to the left branch, and the LSBs of both $X_{(a,i)}$ and $Y_{(a,i)}$ start from the right.

- $X_{(a,i)}[i, j, \cdots, k]$: The $i, j, \cdots, k$ bits of $X_{(a,i)}$.

- $X_{(a,i)}[i : j]$: The bits from i to j of $X_{(a,i)}$, where $i \leq j$.

- w: The number of 32-bit words, i.e., $w = 4$ for a 128-bit block and $w = 8$ for a 256-bit master key.

- R^4: The iteration of 4 rounds of SPECKEY [21, 22] with their corresponding key additions.

- L_w: Linear mixing layer used in SPARX with w-word block size. Thus, L_4 represents the linear mixing layer used in SPARX-128/128 and SPARX-128/256.

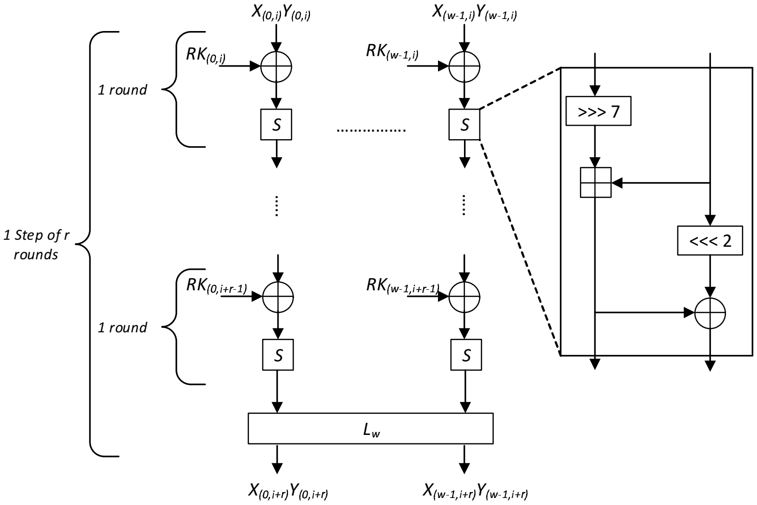


Figure 8.1: SPARX structure

- $\boxplus$: 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.

SPARX [61, 60] is a family of ARX-based Substitution-Permutation Network (SPN) block ciphers. It follows the SPN design construction while using ARX-based S-boxes instead of S-boxes based on look-up tables. The ARX-based S-boxes form a specific category of S-boxes that rely solely on addition, rotation and XOR operations to provide both non-linearity and diffusion. The SPARX family adopts the 32-bit SPECKEY ARX-based S-box (S), shown in Figure 8.1, which resembles one round of SPECK-32 [21, 22] 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.

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), is applied to w words in parallel, and 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. As depicted in Figure 8.1, 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. 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 and the key length of the cipher.

SPARX-128/128 and SPARX-128/256 are two members of the SPARX family which operate on 128-bit blocks using 128-bit and 256-bit keys, respectively. Both variants use 4 rounds in each step and iterate over 8 and 10 steps, i.e., the total number of rounds is 32 and 40, respectively. More precisely, in SPARX-128/128 and SPARX-128/256, 4 SPECKEY S-boxes (S) are iterated simultaneously for 4 times, while being interleaved by the addition of the round keys and then a linear mixing layer (L_4) is applied, as shown in Figure 8.2 which also depicts the structure of L_4.

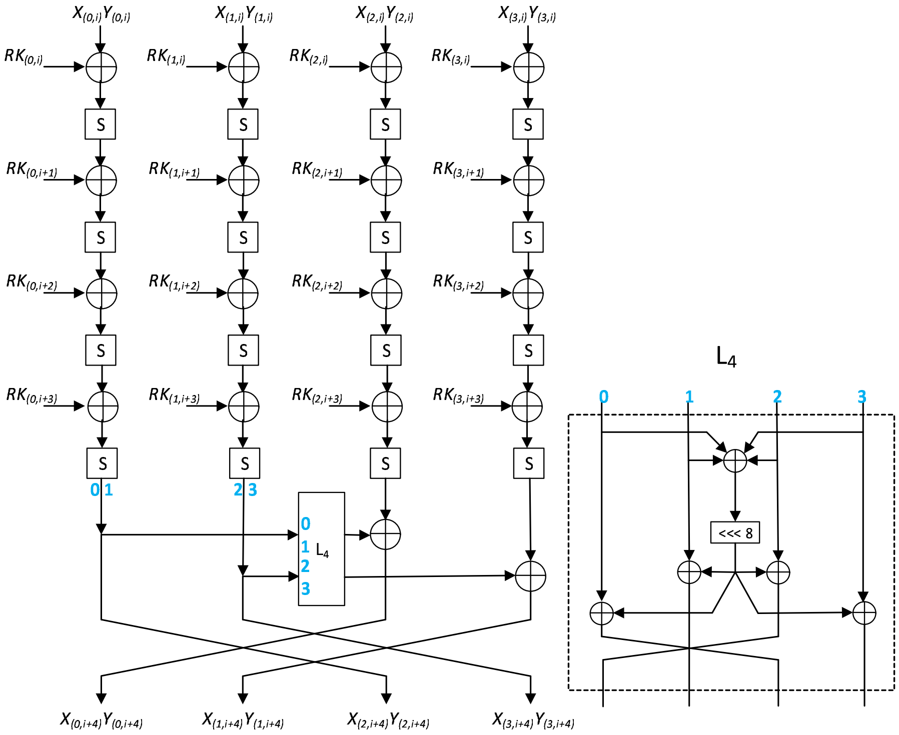


Figure 8.2: SPARX-128/128 and SPARX-128/256 step structure

SPARX-128/128 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 4×32-bit round keys used in branch number 0 of the first step are extracted. Afterwards, the permutation illustrated in Figure 8.3 is applied and then the 4×32-bit round keys used in branch number 1 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 the round keys of a given branch in step j are generated first and then the key state is updated.

SPARX-128/256 key schedule. The 256-bit master key instantiates the key state, denoted by $k_0^0\|k_1^0\|k_2^0\|k_3^0\|k_4^0\|k_5^0\|k_6^0\|k_7^0\|k_8^0\|k_9^0\ \|k_{10}^0\|k_{11}^0\|k_{12}^0\|k_{13}^0\|k_{14}^0\|k_{15}^0$. First, the 4×32-bit

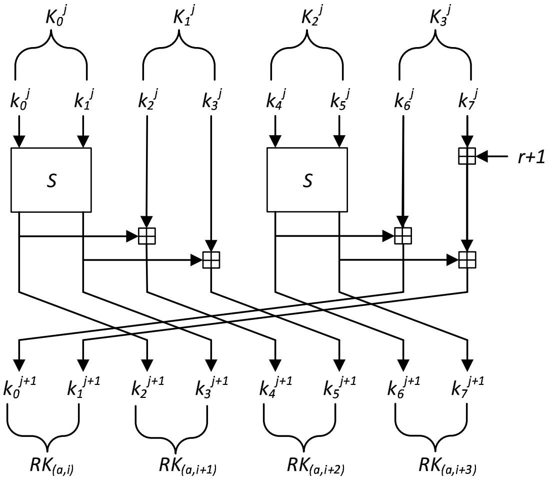


Figure 8.3: SPARX-128/128 key schedule permutation, where the counter r is initialized to 0

round keys used in branch number 0 of the first step are extracted. Then, the 4 × 32-bit round keys used in branch number 1 of the first step are extracted. Afterwards, the permutation illustrated in Figure 8.4 is applied and then the 4 × 32-bit round keys used in branch number 2 and 3 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.

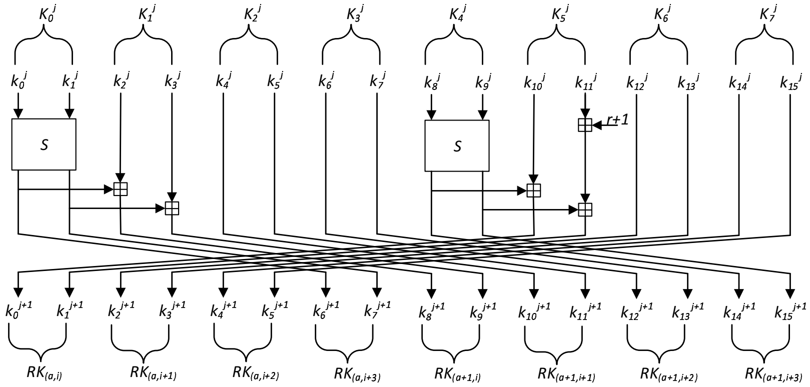


Figure 8.4: SPARX-128/256 key schedule permutation, where the counter r is initialized to 0

8.3 Zero-Correlation Distinguisher of SPARX-128/128 and SPARX-128/256

In this section, we present a 20-round zero-correlation distinguisher for SPARX-128/128 and SPARX-128/256, which will be exploited later in our attacks against 22 rounds (5.5 steps out of 8) of SPARX-128/128 and 24, 25 rounds (6, 6.25 steps out of 10) of SPARX-128/256. As depicted in Figure 8.5, this distinguisher begins with only branch 0 containing a linear mask α_0 at round i. Then, by propagating this linear mask 2 steps forward, and by utilizing Lemma 1 and Lemma 2, we have linear masks 0 and α_4 applied on $X_{(1,i+8)}Y_{(1,i+8)}$ and $X_{(3,i+8)}Y_{(3,i+8)}$, respectively. From the other side, at round $i + 20$, branch 0 has a linear mask β_0, branch 1 has no linear mask, and branch 2 and 3 have linear masks β_1 and β_2, respectively. The linear masks β_1 and β_2 are chosen such that $L_4(\beta_1, \beta_2) = (\beta_0, 0)$. This choice enables us to pass one step backward with only one word having a linear mask β_3 at branch 2. Then, following Lemma 1 and Lemma 2, we can propagate the linear masks backward for one additional step and a linear layer to end with branch 1 and 3 having a non-zero linear mask β_6 and a zero linear mask before applying the inverse of R^4 to obtain $X_{(1,i+8)}Y_{(1,i+8)}$ and $X_{(3,i+8)}Y_{(3,i+8)}$, respectively. Here, R^4 can be considered as a one big S-box, and hence, from Lemma 3, this linear approximation has a zero-correlation.

8.4 Multidimensional Zero-Correlation Cryptanalysis of SPARX-128

The following observations, which stem from the structure of SPARX-128/128 and SPARX-128/256, are exploited in our attacks.

Observation 1 *As depicted in Figure 8.6a, there is a 2-round linear approximation that holds with probability 1 (*$0x0080\ 0x4001 \rightarrow 0x0004\ 0x0004$*).*

Observation 2 *As illustrated in Figure 8.6b, the linear mask* $0\beta\beta0$*, where 0 and* β *denote* $0x0000$ *and 16-bit non-zero linear mask, respectively, propagates through the linear layer* L_4 *as* $\beta\beta00$*, i.e.,* $L_4(0\beta\beta0) = \beta\beta00$*.*

Observation 3 *From Observation 2 and the specification of the S-box, the 20-round distinguisher can be extended to 21-round distinguisher, as shown in Figure 8.6c.*

8.4.1 Multidimensional Zero-Correlation Attack on SPARX-128/256

In this attack, and in order to maximize the number of attacked rounds, we have chosen to place the 20-round distinguisher at the bottom, and add 4 analysis rounds at the top to launch a 24-round attack against SPARX-128/256. Taking into account the key schedule relations, the top 4 analysis rounds involve all the master key bits, and in order to be able to extend 4 rounds above the distinguisher, we utilize Observation 1. In particular, we choose

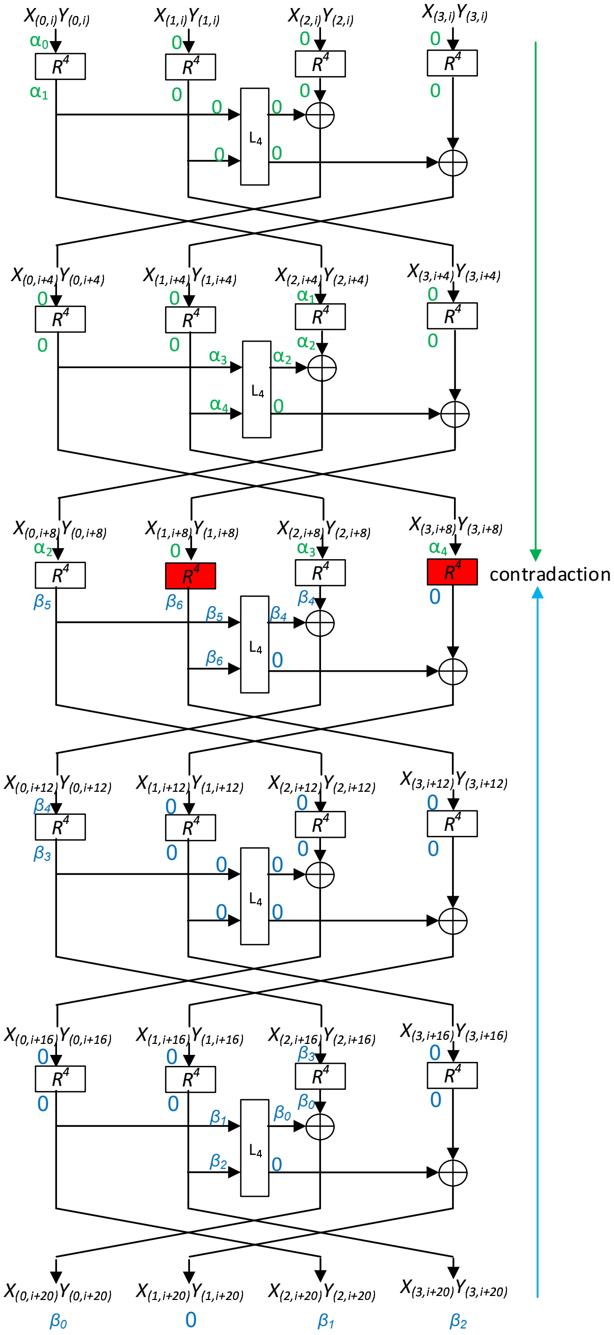
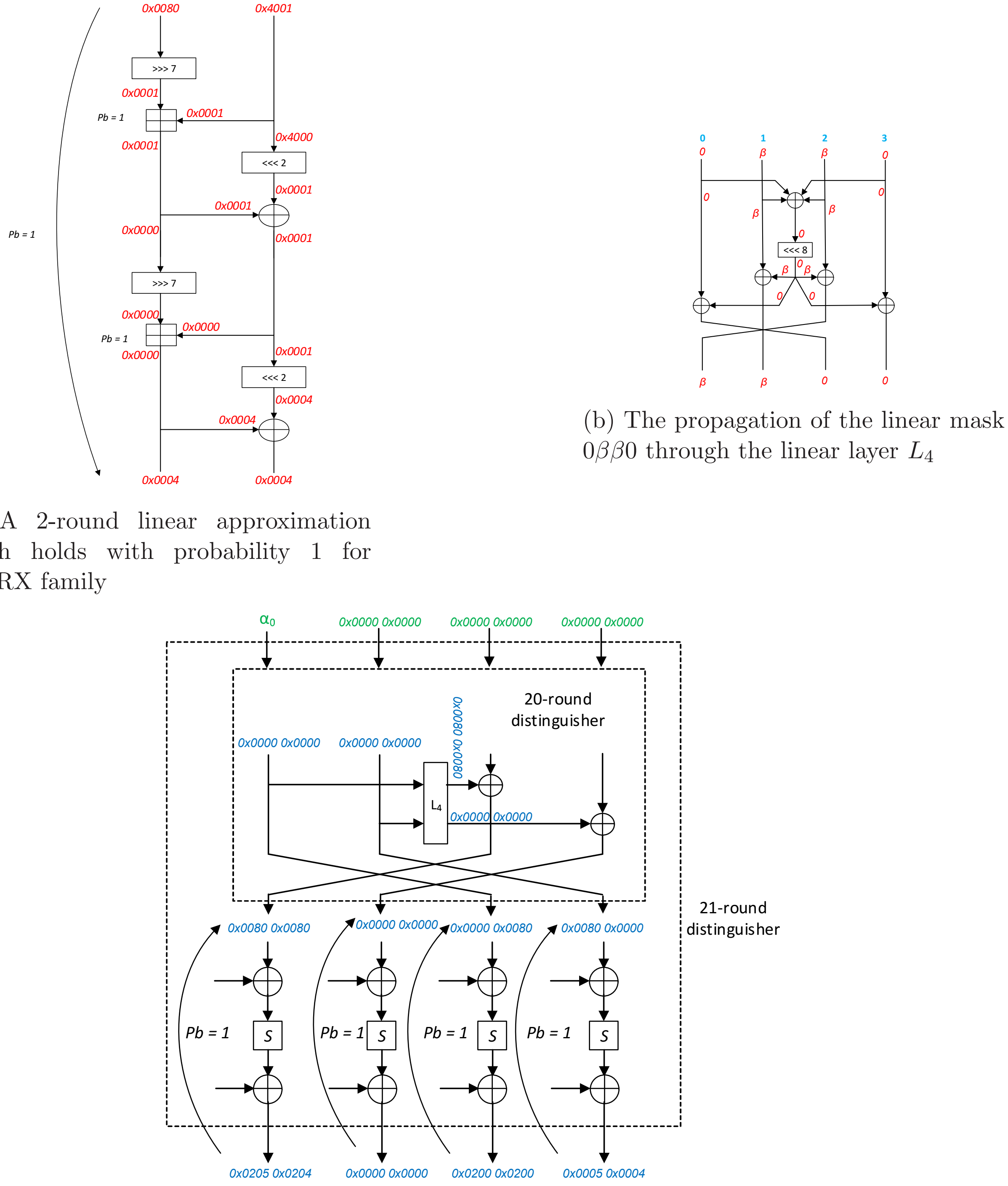


Figure 8.5: A 20-round zero-correlation distinguisher of SPARX-128/128 and SPARX-128/256, where α_i, β_j are 32-bit non-zero linear masks and **0** denotes $0x0000\ 0x0000$ linear mask



(a) A 2-round linear approximation which holds with probability 1 for SPARX family

(b) The propagation of the linear mask $0\beta\beta0$ through the linear layer L_4

(c) A 21-round zero-correlation distinguisher, where α_0 is 32-bit non-zero linear mask

Figure 8.6: Illustrations of Observations 1,2 and 3

a specific linear mask at branch 0 at the beginning of our 20-round zero-correlation distinguisher. This specific linear mask, after propagating it backward through the linear layer L_4, enables us to bypass 2 rounds of branch 0 with probability 1 by exploiting Observation 1 and thus have an extended distinguisher (the dotted one in Figure 8.7).

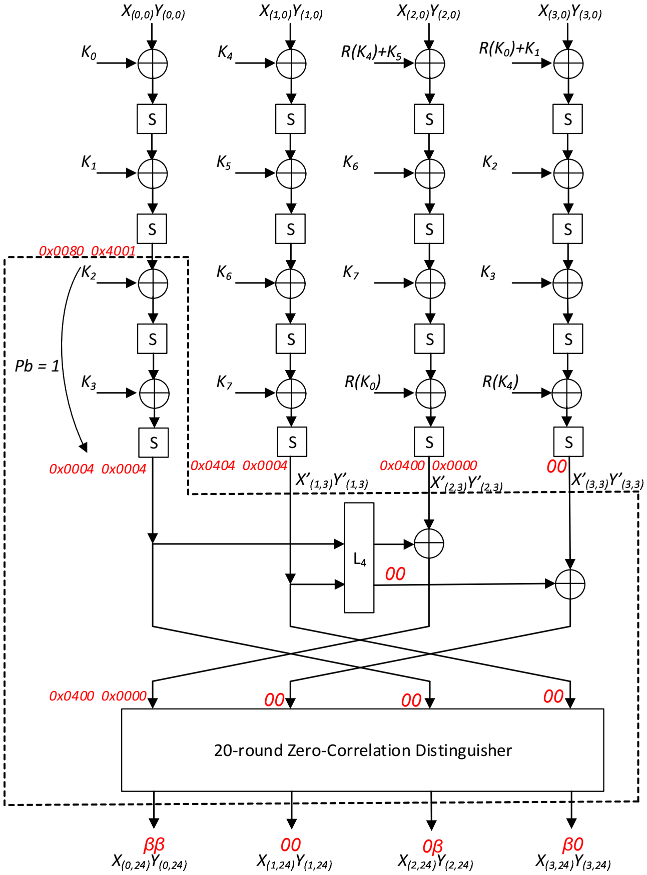


Figure 8.7: A 24-round multidimensional zero-correlation linear cryptanalysis of SPARX-128/256, where 0 and β denotes $0x0000$ and 16-bit non-zero linear mask, respectively

Key Recovery. Here, we chose $\beta = 0x0abc$, where a, b, c are 4-bit non-zero linear masks. Then, the attack proceeds by gathering enough plaintext/ciphertext pairs. Afterwards, we guess the round keys involved in the analysis rounds to estimate the statistic T. However, the complexity of the attack following this strategy exceeds the complexity of exhaustive search. Therefore, we use the partial compression technique in order to reduce the time complexity of the attack as follows:

Step 1. Allocate an array of counters $N_1[X_1]$ and initialize it to zeros, where $X_1 = X_{(0,0)}Y_{(0,0)}||X_{(1,0)}Y_{(1,0)}\ ||X_{(2,0)}Y_{(2,0)}||(X_{(0,24)}[0:11] \oplus Y_{(0,24)}[0:11] \oplus Y_{(2,24)}[0:11] \oplus X_{(3,24)}[0:11])$, i.e., $|X_1| = 108$ bits. Then, from the gathered plaintext/ciphertext pairs compute X_1 and increment the corresponding counter. Since all the non-zero 16-bit linear masks in the ciphertext equal $\beta = 0x0abc$, then, we can store only $(X_{(0,24)}[0:11] \oplus Y_{(0,24)}[0:11] \oplus Y_{(2,24)}[0:$

$11] \oplus X_{(3,24)}[0:11])$ instead of storing each one separately to apply the linear mask β.

Step 2. Allocate an array of counters $N_2[X_2]$ and initialize it to zeros, where $X_2 = X_{(0,0)}Y_{(0,0)}||X_{(1,3)}[0,1,7:15]Y_{(1,3)}[0:10]\ ||X_{(2,0)}Y_{(2,0)}||(X_{(0,24)}[0:11] \oplus Y_{(0,24)}[0:11] \oplus Y_{(2,24)}[0:11] \oplus X_{(3,24)}[0:11])$, i.e., $|X_2| = 98$ bits. Then, guess K_4, K_5, K_6 and partially encrypt X_1 to compute X_2 and add the corresponding counter $N_1[X_1]$ to $N_2[X_2]$.

Step 3. Allocate an array of counters $N_3[X_3]$ and initialize it to zeros, where $X_3 = X_{(0,0)}Y_{(0,0)}||X'_{(1,3)}[2,10]Y'_{(1,3)}[2]\ ||X_{(2,0)}Y_{(2,0)}||(X_{(0,24)}[0:11] \oplus Y_{(0,24)}[0:11] \oplus Y_{(2,24)}[0:11] \oplus X_{(3,24)}[0:11])$, i.e., $|X_3| = 79$ bits. Then, guess 22 bits of K_7 ($K_7[0:10,16,17,23:31] \equiv k_{14}[0,1,7:15], k_{15}[0:10]$) and partially encrypt X_2 to compute X_3 and add the corresponding counter $N_2[X_2]$ to $N_3[X_3]$. Since the linear mask on $X'_{(1,3)}Y'_{(1,3)}$ is $0x0404\ 0x0004$, i.e., we need to compute only 3 bits of $X'_{(1,3)}Y'_{(1,3)}$, and we need only to know 22 bits of $X_{(1,3)}[0,1,7:15]Y_{(1,3)}[0:10]$ and 22 bits of K_7 to compute this linear mask.

Step 4. Allocate an array of counters $N_4[X_4]$ and initialize it to zeros, where $X_4 = X_{(0,0)}Y_{(0,0)}||X'_{(1,3)}[2,10]Y'_{(1,3)}[2]\ ||X_{(2,3)}[0,1,7:15]Y_{(2,3)}[0:10]||(X_{(0,24)}[0:11] \oplus Y_{(0,24)}[0:11] \oplus Y_{(2,24)}[0:11] \oplus X_{(3,24)}[0:11])$, i.e., $|X_4| = 69$ bits. Then, guess the remaining 10 bits of K_7 and partially encrypt X_3 to compute X_4 and add the corresponding counter $N_3[X_3]$ to $N_4[X_4]$.

Step 5. Allocate an array of counters $N_5[X_5]$ and initialize it to zeros, where $X_5 = X_{(0,0)}Y_{(0,0)}||X'_{(1,3)}[2,10]Y'_{(1,3)}[2]\ ||X'_{(2,3)}[10]||(X_{(0,24)}[0:11] \oplus Y_{(0,24)}[0:11] \oplus Y_{(2,24)}[0:11] \oplus X_{(3,24)}[0:11])$, i.e., $|X_5| = 48$ bits. Then, guess 22 bits of $R(K_0)$ ($R(K_0)[0:10,16,17,23:31]$) and partially encrypt X_4 to compute X_5 and add the corresponding counter $N_4[X_4]$ to $N_5[X_5]$.

Step 6. Allocate an array of counters $N_6[X_6]$ and initialize it to zeros, where $X_6 = X_{(0,1)}[0:5,7:15]Y_{(0,1)}[0:14]||X'_{(1,3)}[2,10]Y'_{(1,3)}[2]\ ||X'_{(2,3)}[10]||(X_{(0,24)}[0:11] \oplus Y_{(0,24)}[0:11] \oplus Y_{(2,24)}[0:11] \oplus X_{(3,24)}[0:11])$, i.e., $|X_6| = 46$ bits. Then, guess the remaining 10 bits of $R(K_0)$ and partially encrypt X_5 to compute X_6 and add the corresponding counter $N_5[X_5]$ to $N_6[X_6]$.

Step 7. Allocate an array of counters $N_7[X_7]$ and initialize it to zeros, where $X_7 = X_{(0,2)}[7]Y_{(0,2)}[0,14]||X'_{(1,3)}[2,10]Y'_{(1,3)}[2]\ ||X'_{(2,3)}[10]||(X_{(0,24)}[0:11] \oplus Y_{(0,24)}\ [0:11] \oplus Y_{(2,24)}[0:11] \oplus X_{(3,24)}[0:11])$, i.e., $|X_7| = 19$ bits. Then, guess 30 bits of K_1 ($k_2[0:5,7:15], k_3[0:14]$) and partially encrypt X_6 to compute X_7 and add the corresponding counter $N_6[X_6]$ to $N_7[X_7]$.

The steps of the key recovery phase are summarized in Table 8.1, where the second column gives the keys to be guessed in each step. The third column presents the saved state in each step after the partial encryption, the fourth column is the counter size for each obtained state in the corresponding step, and the fifth column quantifies the time complexity of each step measured in 24-round encryption by considering the number of S-box accesses.

After Step 7, we have guessed 190 key bits (gK) from the master key and evaluated X_7, that contains all the 19 bits involved in computing the zero-correlation masks. Therefore, to

Table 8.1: Key recovery process of the attack on 24-round SPARX-128/256

Step	Guessed keys	Obtained state	Size	Time complexity
1	†	X_1	108	‡
2	K_4, K_5, K_6	X_2	98	$2^{108} \times 2^{3\times 32} \times \frac{3}{24 \times 4} \approx 2^{199}$
3	$K_7[0:10, 16, 17, 23:31]$	X_3	79	$2^{98} \times 2^{96+22} \times \frac{1}{24 \times 4} \approx 2^{209.4}$
4	$K_7[11:15, 18:22]$	X_4	69	$2^{79} \times 2^{118+10} \times \frac{3}{24 \times 4} \approx 2^{202}$
5	$R(K_0)[0:10, 16, 17, 23:31]$	X_5	48	$2^{69} \times 2^{128+22} \times \frac{1}{24 \times 4} \approx 2^{212.4}$
6	$R(K_0)[11:15, 18:22]$	X_6	46	$2^{48} \times 2^{150+10} \times \frac{1}{24 \times 4} \approx 2^{201.4}$
7	$K_1[0:14, 16:21, 23:31]$	X_7	19	$2^{46} \times 2^{160+30} \times \frac{1}{24 \times 4} \approx 2^{229.4}$

†: No additional key guesses needed, ‡: Negligible complexity

recover the master key, the following steps are performed:

1. Allocate an array of counters $V[z]$, where $|z| = 12$ bits.
2. For 2^{19} values of X_7
   (a) Evaluate all 12 basis zero-correlation masks on X_7 and calculate z.
   (b) Update the counter $V[z]$ by $V[z] = V[z] + N_7[X_7]$.
3. For each guessed key gK, compute $T_{gK} = \frac{N \times 2^{12}}{1 - 2^{-12}} \sum_{z=0}^{2^{12}-1} \left(\frac{V[z]}{N} - \frac{1}{2^{12}} \right)^2$.
4. If $T_k < \tau$, then the guessed values of gK are key candidates.
5. Exhaustively search all the remaining key candidates with 2^{66} values for the 66 bits of the key that are not retrieved by the above steps of the attack using 2 plaintext/ciphertext pairs.

Attack complexity. Since the beginning of the distinguisher has a specific linear mask and the end of the distinguisher has a variable 12-bit linear mask β, then $m = 12$, and hence $l = 2^{12} - 1$. Here, we set $\gamma = 2^{-2.7}$ and $\zeta = 2^{-30}$ and hence we have $z_{1-\gamma} \approx 1$ and $z_{1-\zeta} \approx 6$. According to equation (2.2), the data complexity is about $2^{125.5}$ known plaintexts. The total time complexity of the attack encompasses the time complexity of two phases. The first is the time required to reduce the key search space which can be computed from Table 8.1. The second is the time required to retrieve the whole master key by exhaustively searching the remaining $2^{190} \times 2^{-30} = 2^{160}$ key candidates with the 2^{66} key bits not involved in the attack using 2 plaintext/ciphertext pairs. Therefore, the total time complexity of the attack is $2^{229.4} + 2 \times 2^{160} \times 2^{66} \approx 2^{229.65}$ 24-round encryptions.

25-round Zero-Correlation Attack on SPARX-128/256. The above attack can be extended one more round to launch a key recovery attack against 25-round of SPARX-128/256 with the full codebook. This extra round can be obtained by selecting the linear masks at the end of the distinguisher as in Observation 3 to convert the 20-round distinguisher to 21-round distinguisher. However, at this time we will use only one zero-correlation linear approximation. Therefore, we require the full codebook. The time complexity of the attack is dominated by Step 7, and it will be $2^{227.4}$ instead of $2^{229.4}$ because we store only 10 bits instead of 12 bits at the end of the distinguisher.

8.4.2 Multidimensional Zero-Correlation Attack on SPARX-128/128

As depicted in Figure 8.8, in this attack we use the 21-round zero-correlation distinguisher obtained by utilizing Observation 3. Then, we append an additional round at the bottom of the distinguisher. In the previous attack, the analysis rounds were placed above the distinguisher, therefore, the relation of the round keys to the master key was straightforward and we use the master key relations in the attack from the beginning. However, in this attack, we place the analysis round at the bottom of the distinguisher, and hence the relation of the round keys to the master key is not trivial. Therefore, we will perform the attack on the round keys. Then, we will explain how to recover the master key from the recovered round keys. In order to balance the time complexity and the data complexity, we choose α_0 having linear masks in the first 30-bit only.

Key Recovery. Similar to the previous attack, we first gather N plaintext/ciphe-rtext pairs, and then proceed as follows:

Step 1. Allocate an array of counters $N_1[X_1]$ and initialize it to zeros, where $X_1 = X_{(0,0)}[0:13]Y_{(0,0)}[0:15]||X_{(0,22)}[0:13]Y_{(0,22)}[2:13]\ ||X_{(2,22)}[0:4,11]\ Y_{(2,22)}[2:4,11]||X_{(3,22)}[0:13]Y_{(3,22)}[2:13]$, i.e., $|X_1| = 92$ bits. Then, from the N plaintext/ciphertext pairs compute X_1 and increment the corresponding counter.

Step 2. Allocate an array of counters $N_2[X_2]$ and initialize it to zeros, where $X_2 = X_{(0,0)}[0:13]Y_{(0,0)}[0:15]||X_{(0,22)}[0:13]Y_{(0,22)}[2:13]\ ||X_{(2,21)}[9]Y_{(2,21)}[9]$
$||X_{(3,22)}[0:13]Y_{(3,22)}[2:13]$, i.e., $|X_2| = 84$ bits. Then, guess $RK_{(2,22)}[2:4,11,16:20,27]$ and partially decrypt X_1 to compute X_2 and add the corresponding counter $N_1[X_1]$ to $N_2[X_2]$.

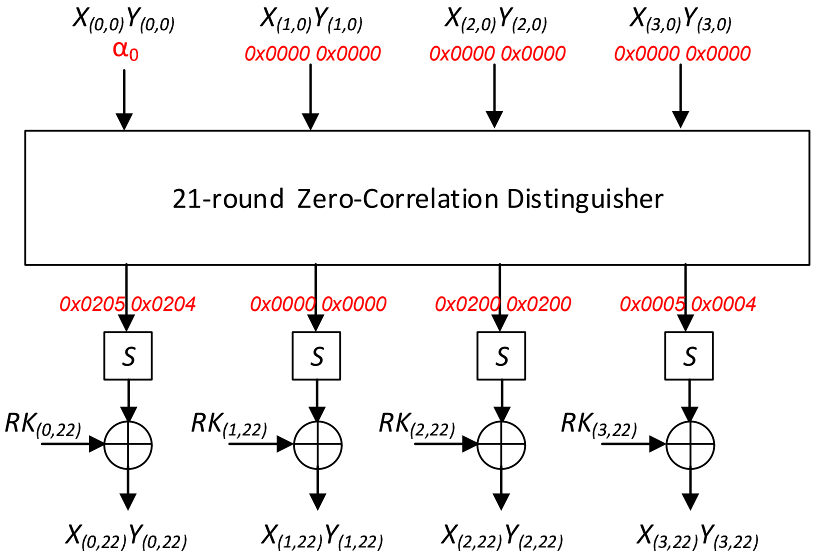


Figure 8.8: A 22-round multidimensional zero-correlation linear cryptanalysis of SPARX-128/128

Step 3. Allocate an array of counters $N_3[X_3]$ and initialize it to zeros, where $X_3 = X_{(0,0)}[0:13]Y_{(0,0)}[0:15]||X_{(0,22)}[0:13]Y_{(0,22)}[2:13]\ ||X_{(2,21)}[9]Y_{(2,21)}[9]$
$||X_{(3,21)}[0,2]Y_{(3,21)}[2]$, i.e., $|X_3| = 61$ bits. Then, guess $RK_{(3,22)}[2:13, 16:29]$ and partially decrypt X_2 to compute X_3 and add the corresponding counter $N_2[X_2]$ to $N_3[X_3]$.
Step 4. Allocate an array of counters $N_4[X_4]$ and initialize it to zeros, where $X_4 = X_{(0,0)}[0:13]Y_{(0,0)}[0:15]||X_{(0,21)}[0,2,9]Y_{(0,21)}[2,9]\ ||X_{(2,21)}[9]Y_{(2,21)}[9]$
$||X_{(3,21)}[0,2]Y_{(3,21)}[2]$, i.e., $|X_4| = 40$ bits. Then, guess $RK_{(0,22)}[2:13, 16:29]$ and partially decrypt X_3 to compute X_4 and add the corresponding counter $N_3[X_3]$ to $N_4[X_4]$.

To determine the surviving round key candidates, we proceed as in the previous attack in section 8.4.1 with $m = 30$, and hence $|z| = 30$ bits. Moreover, instead of using X_7, we use X_4. The number of surviving round key candidates is $2^{62} \times 2^{-\zeta}$. To retrieve the master key, we will, first, retrieve the 128-bit key after applying the key permutation 20 times, i.e., $K_0^{20}||K_1^{20}||K_2^{20}||K_3^{20}$ and, afterwards, we just revert the key schedule permutation 20 times to retrieve the master key. We have retrieved $RK_{(0,22)}[2:13, 16:29]$ which allows us to deduce $K_2^{20}[2:13, 16:29]$, see Figure F.1. Retrieving the remaining 102 bits of $K_0^{20}||K_1^{20}||K_2^{20}||K_3^{20}$ can be done as follows:

1. We guess K_0^{20}, K_3^{20} and the remaining 6 bits of K_2^{20} to compute $RK_{(1,21)}$, $RK_{(1,23)}$, $RK_{(2,21)}, RK_{(2,22)}$. Hence in total we have $2^{62-\zeta+32+32+6-10=122-\zeta}$ remaining key candidates for $K_0^{20}, K_2^{20}, K_3^{20}, RK_{(3,22)}[2:13, 16:29], RK_{(1,21)}$, $RK_{(1,23)}$, $RK_{(2,21)}$, because we have 10-bit filter on $RK_{(2,22)}[2:4, 11, 16:20, 27]$.

2. We guess the remaining 6 bits of $RK_{(3,22)}$ to compute $RK_{(2,20)}, RK_{(1,22)}, K_1^{20}$. Therefore,

in total we have $2^{122-\zeta+6}$ key candidates for $K_0^{20}, K_1^{20}, K_2^{20}, K_3^{20}$.

3. We apply the inverse of the key permutation 20 times to retrieve $2^{122-\zeta+6}$ key candidates for K, i.e., the master key.

4. We test the remaining key candidates using one plaintext/ciphertext pairs to identify the correct key.

Attack complexity. Here, we set $m = 30$ (and hence $l = 2^{30} - 1$), $\gamma = 2^{-2.7}$, and $\zeta = 2^{-26}$. Thus $z_{1-\gamma} \approx 1$ and $z_{1-\zeta} \approx 5.54$. The data complexity is $2^{116.2}$ known plaintexts, which can be computed from equation (2.2). In this case, the total time complexity of the attack is determined by the time complexity of three stages. The first is the time required to reduce the key search space which is dominated by Step 4 and equals $2^{61} \times 2^{10+26+26} \times \frac{1}{22\times 4} \approx 2^{116.54}$. The second is the time required to retrieve the whole master key and equals $2^{62-26+32+32+6} \times \frac{3}{22\times 4} + 2^{122-26+6} \times \frac{2}{22\times 4} + 2^{122-26+6} \times \frac{20\times 2}{22\times 4} + 2^{102} \approx 2^{103}$. The third is the time required by the data collection phase which is equal to $2^{116.2}$. Therefore, the time complexity of the attack is $2^{116.54} + 2^{103} + 2^{116.2} \approx 2^{117.38}$ 22-round encryptions.

Remark: It is worth noting that the above zero-correlation attacks are also applicable to 15 rounds of SPARX-64/128 using the zero-correlation distinguisher shown in Figure G.1 (see also [9]).

8.5 Conclusion

In this chapter, we presented 20 and 21-round zero-correlation distinguishers that are used to launch key recovery attacks against 24, 25 rounds (6, 6.25 out of 10 steps) of SPARX-128/256 and 22 rounds (5.5 out of 8 steps) of SPARX-128/128. To the best of our knowledge these are the first third party attacks against SPARX-128/128 and SPARX-128/256.

Chapter 9

Summary and Future Research Directions

9.1 Summary of contributions

In this thesis, we have evaluated the security of several block ciphers (including Khudra, Piccolo, Kiasu-BC, SKINNY, Midori128, Kuznyechik, and SPARX-128) which employ new design strategies to enhance either the security or performance. In what follows, we briefly summarize the contributions of this thesis.

In chapter 3, the security of two lightweight block ciphers, namely, Khudra and Piccolo have been investigated using the plain MitM attack. By considering the F-function of Khudra as a black box mapping 16-bit to 16-bit (instead of of considering it as 6 rounds and each round has 2 4-bit S-boxes), we are able to find the longest distinguisher for Khudra which covers 6 rounds. Taking into account the key schedule, we choose one of these distinguishers that can be employed in a key recovery attack against 13 rounds of Khudra. Using the same distinguisher to attack 14 rounds of Khudra requires guessing the whole master key. Therefore, we employed another 6-round distinguisher to attack 14 rounds of Khudra. We also analyzed two versions of Piccolo, namely, Piccolo-80 and Piccolo-128. For Piccolo-80, we noticed that the distinguisher that can be built using the smallest number of parameters will not lead to the best attack; and the one that lead to the best attack requires offline computations greater than the exhaustive search. Therefore, we investigated the F-function in order to reduce the number of required parameters. Furthermore, by exploiting the properties of the linear diffusion layer M, we are able to build such distinguisher (that covers 5 rounds) using only 5 parameters. This distinguisher is then extended, by exploiting the key schedule relations, to attack 14 rounds of Piccolo-80. In order to reduce the time and memory complexities of the offline phase, we store a 2^{-7} fraction of the precomputation table

H and repeat the attack for 2^7 times. Since Piccolo-80 and Piccolo-128 have different key schedules, utilizing the same ideas that are exploited on Piccolo-80 will not lead to the best attack. In our attacks on Piccolo-128, we utilized the key dependent sieving technique, in order to build the longest distinguisher. This enables us to build a 7-round distinguisher. Then, this distinguisher is utilized in attacking 16 rounds of Piccolo-128 exploiting the key schedule relations. Using the same distinguisher to attack 17 rounds of Piccolo-128 will render the time complexity of the attack more than the exhaustive search. Therefore, we use another distinguisher that covers 6 rounds, but can be extended to a 17-round attack against Piccolo-128 by exploiting the key schedule relations.

In chapter 4, we investigated the security of a tweakable block cipher, namely, Kiasu-BC, using MitM with efficient enumeration attack. The designers' security claims regarding this cipher are based on the extensive security studies of AES-128, because Kiasu-BC is a tweakable version of AES-128. The best attack on AES-128 covers 7 rounds using the MitM with efficient enumerations. Therefore, the designers of Kiasu-BC concluded that no more than 7 rounds of Kiasu-BC can be attacked using the MitM with efficient enumeration attack. However, we utilized the freedom of the tweak to prove that there is an 8-round attack against Kiasu-BC utilizing the MitM with efficient enumeration attack. The previous distinguisher that are exploited in the 7-round attack against AES-128 covers 4 rounds. Exploiting the tweak, we employed a difference at byte 0 of the tweak equal the b-δ-*set*. This enables us to extend the previous 4-round distinguisher to a 5-round distinguisher. Then, this distinguisher is used to attack 8 rounds of Kiasu-BC. In this attack we utilized the b-δ-*set*, where $b = 5$, instead of δ-*set* to reduce the memory complexity of the attack.

In chapter 5, we investigated the security of a lightweight tweakable block cipher family, namely, SKINNY under the impossible differential attack. While in Kiasu-BC, there are two independent inputs for the tweak and key, in SKINNY there is only one input called tweakey that compromise both the key and the tweak. Therefore, if the tweakey length is t bits and we choose the tweak tk bits then the security margin of the cipher will be $t - tk$ bits. This means that the degree of freedom in the tweak will reduce the security margin of the cipher. Consequently, we chose to focus on the cases where $tk = 0$, and hence the security margin of the cipher will be bounded by t bits. The longest impossible differential distinguisher that begins and ends with only one active byte covers 11 rounds, and there are 16 such extinguishers. All our attacks against all the 6 variants of SKINNY utilize the same distinguisher. We utilized the structure's properties of SKINNY to improve the previous results and present the best attacks against SKINNY in the single-key model. More precisely, we utilized the fact that the tweakey addition are only performed on the first two rows of the state, the binary nature of the MixColumns operations, and the key schedule relations. The previous observations enabled us to launch 18, 20, and 22 rounds of SKINNY-n-n, SKINNY-n-$2n$, and SKINNY-n-$3n$ ($n = 64$ or 128), respectively.

In chapter 6, the security of a low energy consumption block cipher, namely, Midori128 is studied under two different attacks, namely, multiple impossible and truncated differential attacks. Both attacks on Midori128 exploit the binary nature of the MixColumns operation along with the fact that each 8-bit S-box of the four different utilized S-boxes in Midori128 is composed of two 4-bit S-boxes. In the first part of chapter 6, we present the longest impossible differential distinguisher against Midori128 that covers complete 7 rounds including the linear layer of the last round. Then, we exploited four of such distinguishers to launch multiple impossible differential attack against 11 rounds of Midori128 including the pre-whitening and post-whitening keys. This attack enhances the previous results on Midori128 using impossible differential attack. In the second part of chapter 6, by utilizing the previous observations, we are able to minimize the number of active S-boxes considering only the single bit differences in the input and output of the active S-boxes. In order to maintain the single bit difference pattern, we studied the other operations in the round function and then developed an algorithm to find the longest number of rounds that have probability greater than 2^{-128} to act as a distinguisher. Using our algorithm, we found 10 round differential with probability 2^{-118}. Then, using this 10-round distinguisher, we are able to attack 13 rounds of Midori128.

In chapter 7, the security of the standard Russian block cipher Kuznyechik is evaluated using MitM with efficient enumeration technique. The previously best MitM with efficient enumeration attack on this cipher covers 5 rounds and works in the chosen ciphertext model. Here, we showed that there is a 5 round attack using MitM with efficient enumeration in the chosen plaintext model by changing the place of the distinguisher to be on the top instead of the middle as in the previous attack. To extend the 5-round attack by one more round, first, instead of using the structural properties of the linear transformation layer, we identified the exact values of the coefficient of the MDS matrix. Using this matrix, we identified all the relations between set on the inputs and outputs of the linear transformation layer. Second, we exploited one of these relations to present a 3-round distinguisher that optimizes the time complexity of the attack. Then, this distinguisher is placed on the top to launch the attack in the chosen plaintext model. In order to extend 3 rounds below the distinguisher, we performed the matching between the offline and online phases around the linear transformation instead of matching on state byte.

In chapter 8, we studied the security of an ARX-based block cipher, namely, SPARX-128 in the known plaintext model using multidimensional zero-correlation attack. First, we proposed a 20-round distinguisher exploiting the zero-correlation property. By studying the linear transformation layer with the utilized S-box, we found that the 20-round distinguisher can be extended by one more round using specific linear masks at the end of the 20-round distinguisher. To attack 24 rounds of SPARX-128/256, we utilized the 20-round distinguisher and extended it by 4 rounds on the top of it. The top four analysis rounds involve all the master key bits. Therefore, we utilized the fact that there are two-round linear approximation with probability 1. By investigating the key schedule, we chose to place this two-round linear approximation at branch 0. This enables us, using the partial compressing technique, to attack 24 rounds of SPARX-128/256 using 190 involved key bits instead of the whole master key. Then, the 24-round attack is extended one more round to launch a 25-round attack using the 21-round distinguisher instead of the 20-round distinguisher, and as we have specific linear masks at the beginning and end of the distinguisher, the 25-round attack was launched using zero-correlation attack. Therefore, this attack requires the full codebook. Then, we exploited the 21-round distinguisher to attack 22 rounds of SPARX-128/128 using multidimensional zero-correlation. While in the previous attacks on SPARX-128/256 the analysis rounds were chosen on the top of the distinguisher (the relations of the rounds keys to the master key were straightforward), in our attack on SPARX-128/128 the analysis rounds were chosen on the bottom of the distinguisher (in this case, the relations between the rounds keys and the master key are not straightforward). Therefore, we performed the attack on the round keys. Then, from the recovered round keys, we showed how to recover the master key.

9.2 Future work

In what follows, we describe some possible research directions:

- Due to the rapid increase in deploying small computing devices and the difficulty to utilize the current cryptographic primitives (that are designed for desktop applications, and hence exploit their powerful in providing high level of security), the development of lightweight primitives including stream ciphers, block ciphers, hash functions, and authenticated encryption has become one of the main focus of the symmetric-key research community. NIST will open a a call for submissions for lightweight cryptography primitives. It is interesting to investigate how the cryptanalytic approaches presented in this thesis can be utilized to analyze the lightweight symmetric-key primitives that will be submitted to NIST.

- In the context of helping the designers and cryptanalysts to understand the security of the proposed cryptographic primitives, many tools have been developed (e.g., see [123, 64, 109, 121, 140, 122]) to automatically find differential, linear, impossible differential, integral, and zero-correlation distinguishers. All the previously developed tools focus only on finding the longest distinguisher without combining it with the analysis rounds to obtain a complete attack. However, the longest distinguisher may not be the one that lead to the best attack when combined with the analysis rounds. In other words, most of these tools try to model the problem of finding the distinguisher as an optimization problem which has an objective function describing the best distinguisher criteria and set of constrains describing the propagation rules through the distinguisher. Therefore, one interesting research direction is to investigate the automation of finding/optimizing the whole attack instead of just the distinguisher. This goal can be achieved by modeling the whole problem instead of just the distinguisher where the new objective function would compromise both the best distinguisher and attack criteria and the set of constrains describing the propagation rules through both the distinguisher and analysis rounds.

- Post-quantum cryptography focuses on the development of public-key primitives that are not breakable using quantum computers. Current post-quantum algorithms can be categorized as lattice-based cryptosystems, multivariate cryptosystems, hash-based cryptosystems, code-based cryptosystems, and isogeny cryptosystems. It is interesting to investigate if any of the (algebraic) attacks applicable on symmetric-key cryptosystems can be extended to multivariate, code-based, or hash-based post-quantum cryptosystems. The similarities between the two primitives can be seen in their representation where they can be represented using a set of boolean equations or using operations over a small finite field.

Bibliography

[1] GOST 28147-89. Information Processing Systems. Cryptographic Protection. Cryptographic Transformation Algorithm. (In Russian).

[2] NIST National Institute of Standards and Technology, Data Encryption Standard (DES), FIPS-46, 1977.

[3] NIST National Institute of Standards and Technology, Advanced Encryption Standard (AES), FIPS-197, 2001.

[4] 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/GOSTR34112012/GOSTR34112012eng.pdf.

[5] The National Standard of the Russian Federation GOST R 34. -20 . Russian Federal Agency on Technical Regulation and Metrology report, 2015. http://www.tc26.ru/en/standard/draft/ENGGOSTRbsh.pdf.

[6] A. Abdelkhalek, R. AlTawy, M. Tolba, and A. M. Youssef. Meet-in-the-Middle Attacks on Reduced-Round Hierocrypt-3. In K. 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*, pages 187–203. Springer International Publishing, 2015. ISBN 978-3-319-22174-8.

[7] A. Abdelkhalek, Y. Sasaki, Y. Todo, M. Tolba, and A. M. Youssef. MILP Modeling for (Large) S-boxes to Optimize Probability of Differential Characteristics. In *FSE 2018*. Accepted.

[8] A. Abdelkhalek, M. Tolba, and A. M. Youssef. Improved Key Recovery Attack on Round-reduced Hierocrypt-L1 in the Single-Key Setting. In S. R. Chakraborty, P. Schwabe, and J. Solworth, editors, *Security, Privacy, and Applied Cryptography Engineering: 5th International Conference, SPACE 2015, Jaipur, India, October 3-7, 2015, Proceedings*, pages 139–150. Springer International Publishing, 2015. ISBN 978-3-319-24126-5.

[9] 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*, pages 135–146. Springer International Publishing, Cham, 2017. ISBN 978-3-319-57339-7.

[10] S. Ahmadi, Z. Ahmadian, J. Mohajeri, and M. R. Aref. Low-Data Complexity Biclique Cryptanalysis of Block Ciphers With Application to Piccolo and HIGHT. *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY*, 9: 1641–1652, 2014.

[11] Akshima, D. Chang, M. Ghosh, A. Goel, and S. K. Sanadhya. Improved Meet-in-the-Middle Attacks on 7 and 8-Round ARIA-192 and ARIA-256. In A. Biryukov and V. Goyal, editors, *Progress in Cryptology – INDOCRYPT 2015: 16th International Conference on Cryptology in India, Bangalore, India, December 6-9, 2015, Proceedings*, pages 198–217. Springer International Publishing, 2015. ISBN 978-3-319-26617-6.

[12] Akshima, D. Chang, M. Ghosh, A. Goel, and S. K. Sanadhya. Single Key Recovery Attacks on 9-Round Kalyna-128/256 and Kalyna-256/512. In S. Kwon and A. Yun, editors, *Information Security and Cryptology - ICISC 2015: 18th International Conference, Seoul, South Korea, November 25-27, 2015, Revised Selected Papers*, pages 119–135. Springer International Publishing, 2016. ISBN 978-3-319-30840-1.

[13] R. AlTawy, O. Duman, and A. M. Youssef. Fault Analysis of Kuznyechik. IACR Cryptology ePrint Archive, 2015/347, 2015. https://eprint.iacr.org/2015/347.pdf.

[14] R. AlTawy and A. M. Youssef. A Meet in the Middle Attack on Reduced Round Kuznyechik. *IEICE Transactions*, 98-A(10):2194–2198, 2015.

[15] R. Ankele, S. Banik, A. Chakraborti, E. List, F. Mendel, S. M. Sim, and G. Wang. Related-Key Impossible-Differential Attack on Reduced-Round SKINNY. Cryptology ePrint Archive, Report 2016/1127, 2016. http://eprint.iacr.org/2016/1127.

[16] K. Aoki and Y. Sasaki. Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1. In S. Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 70–89. Springer Berlin Heidelberg, 2009.

[17] S. Azimi, Z. Ahmadian, J. Mohajeri, and M. Aref. Impossible differential cryptanalysis of Piccolo lightweight block cipher. In *Information Security and Cryptology (ISCISC), 11th International ISC Conference on*, pages 89–94, Sept 2014.

[18] S. Banik, A. Bogdanov, T. Isobe, K. Shibutani, H. Hiwatari, T. Akishita, and F. Regazzoni. Midori: A Block Cipher for Low Energy. In T. Iwata and J. Cheon, editors, *Advances in Cryptology ASIACRYPT 2015*, volume 9453 of *Lecture Notes in Computer Science*, pages 411–436. Springer Berlin Heidelberg, 2015. ISBN 978-3-662-48799-0.

[19] A. BANNIER, N. BODIN, and E. FILIOL. Automatic Search for a Maximum Probability Differential Characteristic in a Substitution-Permutation Network. Cryptology ePrint Archive, Report 2016/652, 2016. http://eprint.iacr.org/2016/652.

[20] A. Bar-On and N. Keller. A 2^{70} Attack on the Full MISTY1. In M. Robshaw and J. Katz, editors, *Advances in Cryptology – CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, pages 435–456. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. ISBN 978-3-662-53018-4.

[21] 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.

[22] 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.

[23] C. Beierle, J. Jean, S. Klbl, G. Leander, A. Moradi, T. Peyrin, Y. Sasaki, P. Sasdrich, and S. M. Sim. Skinny family of block ciphers: Cryptanalysis competition, 2016.

[24] C. Beierle, J. Jean, S. Kölbl, G. Leander, A. Moradi, T. Peyrin, Y. Sasaki, P. Sasdrich, and S. M. Sim. The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In M. Robshaw and J. Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 123–153. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. ISBN 978-3-662-53008-5.

[25] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. Keccak sponge function family main document, 2010. http://keccak.noekeon.org/Keccak-main-2.1.pdf.

[26] E. Biham, A. Biryukov, and A. Shamir. Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In J. Stern, editor, *Advances in Cryptology - EUROCRYPT 99*, volume 1592 of *Lecture Notes in Computer Science*, pages 12–23. Springer Berlin Heidelberg, 1999. ISBN 978-3-540-65889-4.

[27] 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, pages 254–266. Springer Berlin Heidelberg, 2002. ISBN 978-3-540-36178-7.

[28] E. Biham and A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In A. J. Menezes and S. A. Vanstone, editors, *Advances in Cryptology-CRYPT0’ 90: Proceedings*, pages 2–21. Springer Berlin Heidelberg, 1991. ISBN 978-3-540-38424-3.

[29] E. Biham and A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard*, volume 28. Springer, 1993.

[30] E. Biham and A. Shamir. Differential Cryptanalysis of the Full 16-round DES. In E. F. Brickell, editor, *Advances in Cryptology — CRYPTO’ 92: 12th Annual International Cryptology Conference Santa Barbara, California, USA August 16–20, 1992 Proceedings*, pages 487–496. Springer Berlin Heidelberg, 1993. ISBN 978-3-540-48071-6.

[31] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen. A More Efficient AES Threshold Implementation. In D. Pointcheval and D. Vergnaud, editors, *Progress in Cryptology – AFRICACRYPT 2014*, pages 267–284. Springer International Publishing, 2014. ISBN 978-3-319-06734-6.

[32] A. Biryukov, D. Khovratovich, and L. Perrin. Multiset-Algebraic Cryptanalysis of Reduced Kuznyechik, Khazad, and secret SPNs. *IACR Transactions on Symmetric Cryptology*, 2016(2):226–247, 2017.

[33] A. Biryukov, L. Perrin, and A. Udovenko. Reverse-Engineering the S-Box of Streebog, Kuznyechik and STRIBOBr1. In M. Fischlin and J.-S. Coron, editors, *Advances in Cryptology – EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, volume 9665, pages 372–402. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. ISBN 978-3-662-49890-3.

[34] A. Biryukov, A. Roy, and V. Velichkov. Differential Analysis of Block Ciphers SIMON and SPECK. In C. Cid and C. Rechberger, editors, *Fast Software Encryption: 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, pages 546–570. Springer Berlin Heidelberg, 2015. ISBN 978-3-662-46706-0.

[35] A. Biryukov and V. Velichkov. Automatic Search for Differential Trails in ARX Ciphers. In J. Benaloh, editor, *Topics in Cryptology CT-RSA 2014*, volume 8366 of *Lecture Notes in Computer Science*, pages 227–250. Springer, 2014.

[36] A. Biryukov, V. Velichkov, and Y. Le Corre. Automatic Search for the Best Trails in ARX: Application to Block Cipher Speck. In T. Peyrin, editor, *Fast Software Encryption: 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, pages 289–310. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. ISBN 978-3-662-52993-5.

[37] A. Bogdanov. Analysis and Design of Block Cipher Constructions . Ph.D. Thesis, Ruhr-Universität Bochum, 2009. https://www.emsec.rub.de/media/crypto/attachments/files/2011/04/thesis_andrey.pdf.

[38] A. Bogdanov, D. Chang, M. Ghosh, and S. Sanadhya. Bicliques with Minimal Data and Time Complexity for AES. In J. Lee and J. Kim, editors, *Information Security and Cryptology - ICISC 2014*, volume 8949 of *Lecture Notes in Computer Science*, pages 160–174. Springer International Publishing, 2015. ISBN 978-3-319-15942-3.

[39] A. Bogdanov, H. Geng, M. Wang, L. Wen, and B. Collard. Zero-Correlation Linear Cryptanalysis with FFT and Improved Attacks on ISO Standards Camellia and CLEFIA. In T. Lange, K. Lauter, and P. Lisoněk, editors, *Selected Areas in Cryptography – SAC 2013: 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, pages 306–323. Springer Berlin Heidelberg, 2014. ISBN 978-3-662-43414-7.

[40] A. Bogdanov, D. Khovratovich, and C. Rechberger. Biclique Cryptanalysis of the Full AES. In D. Lee and X. Wang, editors, *Advances in Cryptology ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 344–371. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-25384-3.

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

[42] A. Bogdanov, G. Leander, K. Nyberg, and M. Wang. Integral and Multidimensional Linear Distinguishers with Correlation Zero. In X. Wang and K. Sako, editors, *Advances in Cryptology – ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, pages 244–261. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-34961-4.

[43] A. Bogdanov and C. Rechberger. A 3-Subset Meet-in-the-Middle Attack: Cryptanalysis of the Lightweight Block Cipher KTANTAN. In A. Biryukov, G. Gong, and D. R. Stinson, editors, *Selected Areas in Cryptography*, volume 6544 of *Lecture Notes in Computer Science*, pages 229–240. Springer Berlin Heidelberg, 2010.

[44] J. Borst, L. R. Knudsen, and V. Rijmen. Two Attacks on Reduced IDEA. In W. Fumy, editor, *Advances in Cryptology — EUROCRYPT '97: International Conference on the Theory and Application of Cryptographic Techniques Konstanz, Germany, May 11–15, 1997 Proceedings*, pages 1–13. Springer Berlin Heidelberg, 1997. ISBN 978-3-540-69053-5.

[45] S. Bulygin, M. Walter, and J. Buchmann. Full analysis of PRINTcipher with respect to invariant subspace attack: efficient key recovery and countermeasures. *Designs, Codes and Cryptography*, 73(3):997–1022, 2014.

[46] C. Cannière, O. Dunkelman, and M. Knežević. KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers. In C. Clavier and K. Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*, pages 272–288. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-04137-2.

[47] A. Canteaut, T. Fuhr, H. Gilbert, M. Naya-Plasencia, and J.-R. Reinhard. Multiple differential cryptanalysis of round-reduced PRINCE. In C. Cid and C. Rechberger, editors, *Fast Software Encryption - FSE 2014*, volume 8540 of *Lecture Notes in Computer Science*, pages 591–610. Springer Berlin Heidelberg, 2015.

[48] A. Canteaut, M. Naya-Plasencia, and B. Vayssière. Sieve-in-the-Middle: Improved MITM Attacks. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 222–240. Springer-Verlag Berlin Heidelberg, 2013.

[49] M. Çoban, F. Karakoç, and Özkan Boztaş. Biclique Cryptanalysis of TWINE. In J. Pieprzyk, A.-R. Sadeghi, and M. Manulis, editors, *Cryptology and Network Security*, volume 7712 of *Lecture Notes in Computer Science*, pages 43–55. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-35404-5.

[50] D. Chaum and J.-H. Evertse. Cryptanalysis of DES with a reduced number of rounds; sequence of linear factors in block ciphers. In W. HC, editor, *Advances in cryptology CRYPTO85*, volume 218 of *Lecture Notes in Computer Science*, pages 192–211. Springer, Berlin, 1986.

[51] Z. Chen, H. Chen, and X. Wang. Cryptanalysis of Midori128 Using Impossible Differential Techniques. In F. Bao, L. Chen, R. H. Deng, and G. Wang, editors, *Information Security Practice and Experience*, pages 1–12. Springer, 2016. ISBN 978-3-319-49151-6.

[52] J. Daemen, L. 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*, pages 149–165. Springer Berlin Heidelberg, 1997. ISBN 978-3-540-69243-0.

[53] 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*, pages 222–238. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-45325-3.

[54] J. Daemen and V. Rijmen. *The Design of Rijndael*. Springer-Verlag New York, Inc., 2002. ISBN 3540425802.

[55] 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*, pages 116–126. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-71039-4.

[56] H. Demirci, İ. Taşkın, M. Çoban, and A. Baysal. Improved Meet-in-the-Middle Attacks on AES. In B. 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*, pages 144–156. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-10628-6.

[57] P. Derbez and P.-A. Fouque. Exhausting Demirci-Selçuk Meet-in-the-Middle Attacks Against Reduced-Round AES. In S. Moriai, editor, *Fast Software Encryption: 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424, pages 541–560. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-662-43933-3.

[58] P. Derbez, P.-A. 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*, pages 371–387. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-38348-9.

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

[60] D. Dinu, L. Perrin, A. Udovenko, V. Velichkov, J. Groschdl, 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.

[61] D. Dinu, L. Perrin, A. Udovenko, V. Velichkov, J. Großschädl, and A. Biryukov. Design Strategies for ARX with Provable Bounds: SPARX and LAX. 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*, pages 484–513. Springer Berlin Heidelberg, 2016. ISBN 978-3-662-53887-6.

[62] C. Dobraunig, M. Eichlseder, and F. Mendel. Square Attack on 7-Round Kiasu-BC. In M. Manulis, A.-R. Sadeghi, and S. Schneider, editors, *Applied Cryptography and Network Security: 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings*, pages 500–517. Springer International Publishing, Cham, 2016. ISBN 978-3-319-39555-5.

[63] 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*, pages 158–176. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-17373-8.

[64] D. Gérault and P. Lafourcade. Related-Key Cryptanalysis of Midori. In O. Dunkelman and S. K. Sanadhya, editors, *Progress in Cryptology – INDOCRYPT 2016*, pages 287–304. Springer, 2016. ISBN 978-3-319-49890-4.

[65] L. Grassi, C. Rechberger, , and S. Rnjom. Subspace Trail Cryptanalysis and its Applications to AES. Cryptology ePrint Archive, Report 2016/592, 2016. http://eprint.iacr.org/2016/592.

[66] J. Guo, J. Jean, I. Nikolić, K. Qiao, Y. Sasaki, and S. M. Sim. Invariant Subspace Attack Against Full Midori64. Cryptology ePrint Archive, Report 2015/1189, 2015. http://eprint.iacr.org/2015/1189.

[67] J. Guo, J. Jean, I. Nikolić, and Y. Sasaki. Meet-in-the-Middle Attacks on Generic Feistel Constructions. In P. Sarkar and T. Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 458–477. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-662-45611-8.

[68] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B.-S. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, and S. Chee. HIGHT: A New Block Cipher Suitable for Low-resource Device. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 46–59. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 3-540-46559-6, 978-3-540-46559-1.

[69] T. Isobe and K. Shibutani. Security Analysis of the Lightweight Block Ciphers XTEA, LED and Piccolo. In W. Susilo, Y. Mu, and J. Seberry, editors, *Information Security and Privacy*, volume 7372 of *Lecture Notes in Computer Science*, pages 71–86. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-31447-6.

[70] T. Jakobsen and L. R. Knudsen. The interpolation attack on block ciphers. In E. Biham, editor, *Fast Software Encryption: 4th International Workshop, FSE'97 Haifa, Israel, January 20–22 1997 Proceedings*, pages 28–40. Springer Berlin Heidelberg, 1997. ISBN 978-3-540-69243-0.

[71] J. Jean, I. Nikolić, and T. Peyrin. KIASU. Submission to the CAESAR competition, 2014. http://competitions.cr.yp.to/round1/kiasuv1.pdf.

[72] J. Jean, I. Nikolić, 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*, pages 274–288. Springer Berlin Heidelberg, 2014. ISBN 978-3-662-45608-8.

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

[74] K. Jeong. Cryptanalysis of block cipher Piccolo suitable for cloud computing. *The Journal of Supercomputing*, 66(2):829–840, 2013.

[75] K. Jeong, H. Kang, C. Lee, J. Sung, and S. Hong. Biclique Cryptanalysis of Lightweight Block Ciphers PRESENT, Piccolo and LED. IACR Cryptology ePrint Archive, 2012/621, 2012. https://eprint.iacr.org/2012/621.pdf.

[76] F. Karakoç, Ö. M. Sağdıçoğlu, M. E. Gönen, and O. Ersoy. Impossible Differential Cryptanalysis of 16/18-Round Khudra. In A. Bogdanov, editor, *Lightweight Cryptography for Security and Privacy: 5th International Workshop, LightSec 2016, Aksaray, Turkey, September 21-22, 2016, Revised Selected Papers*, pages 33–44. Springer International Publishing, Cham, 2017.

[77] J. Kim, S. Hong, S. Lee, J. Song, and H. Yang. Truncated Differential Attacks on 8-Round CRYPTON. In J.-I. Lim and D.-H. Lee, editors, *Information Security and Cryptology - ICISC 2003: 6th International Conference, Seoul, Korea, November 27-28, 2003. Revised Papers*, pages 446–456. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-24691-6.

[78] L. Knudsen, G. Leander, A. Poschmann, and M. Robshaw. PRINTcipher: A Block Cipher for IC-Printing. In S. Mangard and F.-X. Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 16–32. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15030-2.

[79] L. Knudsen and D. 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*, pages 112–127. Springer Berlin Heidelberg, 2002. ISBN 978-3-540-45661-2.

[80] L. R. Knudsen. Truncated and higher order differentials. In B. Preneel, editor, *Fast Software Encryption: Second International Workshop Leuven, Belgium, December 14–16, 1994 Proceedings*, pages 196–211. Springer Berlin Heidelberg, 1995. ISBN 978-3-540-47809-6.

[81] Knudsen, Lars R. and Robshaw, M. J. B. and Wagner, David. Truncated Differentials and Skipjack. In M. Wiener, editor, *Advances in Cryptology — CRYPTO' 99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings*, pages 165–180. Springer Berlin Heidelberg, 1999. ISBN 978-3-540-48405-9.

[82] S. Kolay and D. Mukhopadhyay. Khudra: A New Lightweight Block Cipher for FPGAs. In R. S. Chakraborty, V. Matyas, and P. Schaumont, editors, *Security, Privacy, and Applied Cryptography Engineering*, volume 8804 of *Lecture Notes in Computer Science*, pages 126–145. Springer International Publishing, 2014.

[83] T. Koyama, L. Wang, Y. Sasaki, K. Sakiyama, and K. Ohta. New Truncated Differential Cryptanalysis on 3D Block Cipher. In M. D. Ryan, B. Smyth, and G. Wang, editors, *Information Security Practice and Experience: 8th International Conference, ISPEC 2012, Hangzhou, China, April 9-12, 2012. Proceedings*, pages 109–125. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-29101-2.

[84] X. Lai. Higher Order Derivatives and Differential Cryptanalysis. In R. E. Blahut, D. J. Costello, U. Maurer, and T. Mittelholzer, editors, *Communications and Cryptography: Two Sides of One Tapestry*, pages 227–233. Springer US, 1994. ISBN 978-1-4615-2694-0.

[85] S. K. Langford and M. E. Hellman. Differential-Linear Cryptanalysis. In Y. G. Desmedt, editor, *Advances in Cryptology — CRYPTO '94: 14th Annual International Cryptology Conference Santa Barbara, California, USA August 21–25, 1994 Proceedings*, pages 17–25. Springer Berlin Heidelberg, 1994. ISBN 978-3-540-48658-9.

[86] G. Leander, M. A. Abdelraheem, H. AlKhzaimi, and E. Zenner. A Cryptanalysis of PRINTcipher: The Invariant Subspace Attack. In P. Rogaway, editor, *Advances in Cryptology – CRYPTO 2011: 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 206–221. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-22792-9.

[87] G. Leander, B. Minaud, and S. Rønjom. A Generic Approach to Invariant Subspace Attacks: Cryptanalysis of Robin, iSCREAM and Zorro. 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*, pages 254–283. Springer Berlin Heidelberg, 2015. ISBN 978-3-662-46800-5.

[88] G. Leander, C. Paar, A. Poschmann, and K. Schramm. New Lightweight DES Variants. In A. Biryukov, editor, *Fast Software Encryption*, volume 4593 of *Lecture Notes in Computer Science*, pages 196–210. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-74617-1.

[89] S. Lee, S. Hong, S. Lee, J. Lim, and S. Yoon. Truncated Differential Cryptanalysis of Camellia. In K. Kim, editor, *Information Security and Cryptology — ICISC 2001: 4th International Conference Seoul, Korea, December 6–7,2001 Proceedings*, pages 32–38. Springer Berlin Heidelberg, 2002. ISBN 978-3-540-45861-6.

[90] G. Leurent. Improved Differential-Linear Cryptanalysis of 7-Round Chaskey with Partitioning. In M. Fischlin and J.-S. Coron, editors, *Advances in Cryptology – EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, pages 344–371. Springer Berlin Heidelberg, 2016. ISBN 978-3-662-49890-3.

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

[92] L. Li, K. Jia, X. Wang, and X. Dong. Meet-in-the-Middle Technique for Truncated Differential and Its Applications to CLEFIA and Camellia. In G. Leander, editor, *Fast Software Encryption: 22nd International Workshop, FSE 2015, Istanbul, Turkey,*

March 8-11, 2015, Revised Selected Papers, pages 48–70. Springer Berlin Heidelberg, 2015. ISBN 978-3-662-48116-5.

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

[94] C. Lim and T. Korkishko. mCrypton A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors. In J.-S. Song, T. Kwon, and M. Yung, editors, *Information Security Applications*, volume 3786 of *Lecture Notes in Computer Science*, pages 243–258. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-31012-9.

[95] L. Lin and W. Wu. Meet-in-the-Middle Attacks on Reduced-Round Midori-64. IACR Cryptology ePrint Archive, 2015/1165, 2015. https://eprint.iacr.org/2015/1165.pdf.

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

[97] G. Liu, M. Ghosh, and L. Song. Security Analysis of SKINNY under Related-Tweakey Settings. Cryptology ePrint Archive, Report 2016/1108, 2016. http://eprint.iacr.org/2016/1108.

[98] Y. Liu, Q. Wang, and V. Rijmen. Automatic Search of Linear Trails in ARX with Applications to SPECK and Chaskey. In M. Manulis, A.-R. Sadeghi, and S. Schneider, editors, *Applied Cryptography and Network Security: 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings*, pages 485–499. Springer International Publishing, 2016. ISBN 978-3-319-39555-5.

[99] J. Lu. Cryptanalysis of Block Ciphers . Ph.D. Thesis, Royal Holloway, University of London, 2008. https://www.ma.rhul.ac.uk/static/techrep/2008/RHUL-MA-2008-19.pdf.

[100] J. Lu, Y. Wei, J. Kim, and E. Pasalic. The Higher-Order Meet-in-the-Middle Attack and Its Application to the Camellia Block Cipher. In S. Galbraith and M. Nandi, editors, *Progress in Cryptology - INDOCRYPT 2012: 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*, pages 244–264. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-34931-7.

[101] X. Ma and K. Qiao. Related-Key Rectangle Attack on Round-reduced Khudra Block Cipher. In M. Qiu, S. Xu, M. Yung, and H. Zhang, editors, *Network and System Security: 9th International Conference, NSS 2015, New York, NY, USA, November 3-5, 2015, Proceedings*, pages 331–344. Springer International Publishing, Cham, 2015.

[102] M. Matsui. Linear Cryptanalysis Method for DES Cipher. In T. Helleseth, editor, *Advances in Cryptology EUROCRYPT 93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer Berlin Heidelberg, 1994.

[103] M. Matsui and A. Yamagishi. A New Method for Known Plaintext Attack of FEAL Cipher. In R. Rueppel, editor, *Advances in Cryptology – EUROCRYPT 92*, volume 658 of *Lecture Notes in Computer Science*, pages 81–91. Springer Berlin Heidelberg, 1993. ISBN 978-3-540-56413-3.

[104] 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*, pages 260–276. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-03317-9.

[105] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot. *Handbook of Applied Cryptography*, chapter 7, pages 223–282. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996. ISBN 0849385237.

[106] M. Minier. On the Security of Piccolo Lightweight Block Cipher against Related-Key Impossible Differentials. In G. Paul and S. Vaudenay, editors, *Progress in Cryptology INDOCRYPT 2013*, volume 8250 of *Lecture Notes in Computer Science*, pages 308–318. Springer International Publishing, 2013. ISBN 978-3-319-03514-7.

[107] S. Moriai, M. Sugita, K. Aoki, and M. Kanda. Security of E2 against Truncated Differential Cryptanalysis. In H. Heys and C. Adams, editors, *Selected Areas in Cryptography: 6th Annual International Workshop, SAC'99 Kingston, Ontario, Canada, August 9–10, 1999 Proceedings*, pages 106–117. Springer Berlin Heidelberg, 2000. ISBN 978-3-540-46513-3.

[108] N. Mouha, V. Velichkov, C. D. Cannire, and B. Preneel. The Differential Analysis of S-Functions. In A. Biryukov, G. Gong, and D. R. Stinson, editors, *Selected Areas in Cryptography*, volume 6544 of *Lecture Notes in Computer Science*, pages 36–56. Springer, 2011.

[109] N. Mouha, Q. Wang, D. Gu, and B. Preneel. Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In C.-K. Wu, M. Yung, and D. Lin, editors, *Information Security and Cryptology*, volume 7537 of *Lecture Notes in Computer Science*, pages 57–76. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-34703-0.

[110] K. Nyberg and L. R. Knudsen. Provable security against a differential attack. *Journal of Cryptology*, 8(1):27–37, 1995.

[111] R. Oliynykov, I. Gorbenko, O. Kazymyrov, V. Ruzhentsev, O. Kuznetsov, Y. Gorbenko, A. Boiko, O. Dyrda, V. Dolgov, and A. Pushkaryov. A New Standard of Ukraine: The Kupyna Hash Function . IACR Cryptology ePrint Archive, 2015/885, 2015. https://eprint.iacr.org/2015/885.pdf.

[112] T. Peyrin and Y. Seurin. Counter-in-Tweak: Authenticated Encryption Modes for Tweakable Block Ciphers. In M. Robshaw and J. Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 33–63. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. ISBN 978-3-662-53018-4.

[113] S. Rasoolzadeh, Z. Ahmadian, M. Salmasizadeh, and M. R. Aref. An Improved Truncated Differential Cryptanalysis of KLEIN. IACR Cryptology ePrint Archive, Report 2014/485, 2014. https://eprint.iacr.org/2014/485.pdf.

[114] S. Sadeghi, T. Mohammadi, and N. Bagheri. Cryptanalysis of Reduced round SKINNY Block Cipher. Cryptology ePrint Archive, Report 2016/1120, 2016. http://eprint.iacr.org/2016/1120.

[115] Y. Sasaki and Y. Todo. New Impossible Differential Search Tool from Design and Cryptanalysis Aspects. Cryptology ePrint Archive, Report 2016/1181, 2016. http://eprint.iacr.org/2016/1181.

[116] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai. Piccolo: An Ultra-Lightweight Blockcipher. In B. Preneel and T. Takagi, editors, *Cryptographic Hardware and Embedded Systems CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 342–357. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-23950-2.

[117] V. Shishkin, D. Dygin, I. Lavrikov, G. Marshalko, V. Rudskoy, and D. Trifonov. *Low-Weight and Hi-End: Draft Russian Encryption Standard*, pages 183–188. 2014.

[118] J. Song, K. Lee, and H. Lee. Biclique Cryptanalysis on Lightweight Block Cipher: HIGHT and Piccolo. *Int. J. Comput. Math.*, 90(12):2564–2580, 2013.

[119] L. Song, Z. Huang, and Q. Yang. Automatic Differential Analysis of ARX Block Ciphers with Application to SPECK and LEA. In K. J. Liu and R. Steinfeld, editors, *Information Security and Privacy: 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part II*, pages 379–394. Springer International Publishing, 2016. ISBN 978-3-319-40367-0.

[120] L. Sun, K. Fu, and M. Wang. Improved Zero-Correlation Cryptanalysis on SIMON. In D. Lin, X. Wang, and M. Yung, editors, *Information Security and Cryptology: 11th*

International Conference, Inscrypt 2015, Beijing, China, November 1-3, 2015, Revised Selected Papers, pages 125–143. Springer International Publishing, Cham, 2016. ISBN 978-3-319-38898-4.

[121] 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.

[122] L. Sun, W. Wang, and M. Wang. Automatic Search of Bit-Based Division Property for ARX Ciphers and Word-Based Division Property. Cryptology ePrint Archive, Report 2017/860, 2017. http://eprint.iacr.org/2017/860.

[123] S. Sun, D. Gerault, P. Lafourcade, Q. Yang, Y. Todo, K. Qiao, and L. Hu. Analysis of AES, SKINNY, and Others with Constraint Programming. Cryptology ePrint Archive, Report 2017/162, 2017. http://eprint.iacr.org/2017/162.

[124] 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. IACR Cryptology ePrint Archive, Report 2014/747, 2014. https://eprint.iacr.org/2014/747.pdf.

[125] Y. Todo. Integral Cryptanalysis on Full MISTY1. In R. Gennaro and M. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 413–432. Springer Berlin Heidelberg, 2015. ISBN 978-3-662-47989-6.

[126] M. Tolba, A. Abdelkhalek, and A. M. Youssef. Multidimensional Zero-Correlation Linear Cryptanalysis of Reduced Round SPARX-128. In *SAC 2017*. To appear in Springer LNCS.

[127] M. Tolba, A. Abdelkhalek, and A. M. Youssef. Meet-in-the-Middle Attacks on Round-Reduced Khudra. In S. R. Chakraborty, P. Schwabe, and J. Solworth, editors, *Security, Privacy, and Applied Cryptography Engineering: 5th International Conference, SPACE 2015, Jaipur, India, October 3-7, 2015, Proceedings*, pages 127–138. Springer International Publishing, 2015. ISBN 978-3-319-24126-5.

[128] M. Tolba, A. Abdelkhalek, and A. M. Youssef. A Meet in the Middle Attack on Reduced Round Kiasu-BC . *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, E99-A(10):1888–1890, 2016.

[129] 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*, pages 3–20. Springer International Publishing, 2016. ISBN 978-3-319-29078-2.

[130] 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*, pages 3–17. Springer International Publishing, 2016. ISBN 978-3-319-45871-7.

[131] 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*, pages 117–134. Springer International Publishing, Cham, 2017. ISBN 978-3-319-57339-7.

[132] M. Tolba, A. Abdelkhalek, and A. M. Youssef. Improved Multiple Impossible Differential Cryptanalysis of Midori128 . *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, E100-A(8):1733–1737, 2017.

[133] M. Tolba and A. M. Youssef. Improved Meet-in-the-Middle Attacks on Reduced Round Kuznyechik. In *ICISC 2017*. To appear in Springer LNCS.

[134] M. Tolba and A. M. Youssef. Generalized MitM attacks on full TWINE. *Information Processing Letters*, 116(2):128 – 135, 2016.

[135] Y. Wang and W. Wu. Improved Multidimensional Zero-Correlation Linear Cryptanalysis and Applications to LBlock and TWINE. In W. Susilo and Y. Mu, editors, *Information Security and Privacy: 19th Australasian Conference, ACISP 2014, Wollongong, NSW, Australia, July 7-9, 2014. Proceedings*, pages 1–16. Springer International Publishing, Cham, 2014. ISBN 978-3-319-08344-5.

[136] Y. Wang, W. Wu, and X. Yu. Biclique Cryptanalysis of Reduced-Round Piccolo Block Cipher. In M. Ryan, B. Smyth, and G. Wang, editors, *Information Security Practice and Experience*, volume 7232 of *Lecture Notes in Computer Science*, pages 337–352. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-29100-5.

[137] Y. Wang, W. Wu, X. Yu, and L. Zhang. Security on LBlock against Biclique Cryptanalysis. In D. H. Lee and M. Yung, editors, *WISA*, volume 7690 of *Lecture Notes*

in Computer Science, pages 1–14. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-35415-1.

[138] L. Wen, M. Wang, A. Bogdanov, and H. Chen. Multidimensional zero-correlation attacks on lightweight block cipher HIGHT: Improved cryptanalysis of an ISO standard. *Information Processing Letters*, 114(6):322 – 330, 2014.

[139] H. Wu, F. Bao, R. H. Deng, and Q. Z. Ye. Improved Truncated Differential Attacks on SAFER. In K. Ohta and D. Pei, editors, *Advances in Cryptology — ASIACRYPT'98: International Conference on the Theory and Application of Cryptology and Information Security Beijing, China, October 18–22, 1998 Proceedings*, pages 133–147. Springer Berlin Heidelberg, 1998. ISBN 978-3-540-49649-6.

[140] 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*, pages 648–678. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[141] H. Xu, P. Jia, G. Huang, and X. Lai. Multidimensional Zero-Correlation Linear Cryptanalysis on 23-Round LBlock-s. In S. Qing, E. Okamoto, K. Kim, and D. Liu, editors, *Information and Communications Security: 17th International Conference, ICICS 2015, Beijing, China, December 9-11, 2015, Revised Selected Papers*, pages 97–108. Springer International Publishing, Cham, 2016. ISBN 978-3-319-29814-6.

[142] Q. Yang, L. Hu, S. Sun, and L. Song. Related-Key Impossible Differential Analysis of Full Khudra. In K. Ogawa and K. Yoshioka, editors, *Advances in Information and Computer Security: 11th International Workshop on Security, IWSEC 2016, Tokyo, Japan, September 12-14, 2016, Proceedings*, pages 135–146. Springer International Publishing, Cham, 2016. ISBN 978-3-319-44524-3.

[143] B. Zhu and G. Gong. Multidimensional meet-in-the-middle attack and its applications to KATAN32/48/64. IACR Cryptology ePrint Archive, 2011/619, 2011. https://eprint.iacr.org/2011/619.pdf.

Appendix A

SKINNY-64-128 Key Schedule Relations

Tables A.1, A.2 illustrate the tweakey and equivalent tweakey relations that are considered in the analysis rounds. We have 28 tweakey nibbles and 10 equivalent tweakey nibbles that are used in the analysis rounds. In this appendix, by utilizing the properties of the tweakey schedule, we show that these tweakey and equivalent tweakey nibbles have only 2^{116} possible values.

For the tweakey nibbles $TK_{17}[t]$ and $TK_{19}[f]$, the following relations hold:

$$\begin{array}{ll}
TK_{17}[t][0] = \mathit{TK1}[l][0] \oplus \mathit{TK2}[l]\{0,1,2,3\} & TK_{19}[f][0] = \mathit{TK1}[l][0] \oplus \mathit{TK2}[l]\{0,1,3\} \\
TK_{17}[t][1] = \mathit{TK1}[l][1] \oplus \mathit{TK2}[l]\{0,1,2\} & TK_{19}[f][1] = \mathit{TK1}[l][1] \oplus \mathit{TK2}[l]\{0,1,2,3\} \\
TK_{17}[t][2] = \mathit{TK1}[l][2] \oplus \mathit{TK2}[l]\{1,2,3\} & TK_{19}[f][2] = \mathit{TK1}[l][2] \oplus \mathit{TK2}[l]\{0,1,2\} \\
TK_{17}[t][3] = \mathit{TK1}[l][3] \oplus \mathit{TK2}[l]\{0,2\} & TK_{19}[f][3] = \mathit{TK1}[l][3] \oplus \mathit{TK2}[l]\{1,2,3\},
\end{array}$$

for $t = 0,1,2,3,4,5,6$, $f = 2,0,4,7,6,3,5$ and $l = 9,15,8,13,10,14,12$, respectively. From the above relations we can deduce $\mathit{TK1}[l]$, $\mathit{TK2}[l]$. Therefore, we have $2^{2\times7\times4=56}$ possible values for these 14 nibbles. Moreover, the knowledge of $\mathit{TK1}[e]$, $\mathit{TK2}[e]$, where $e = 13,14,15$ allows us to deduce the values of $ETK_2[7,10,13]$, and the knowledge of of $\mathit{TK1}[10]$, $\mathit{TK2}[10]$ allows us to deduce the value of $TK_{15}[2]$. In addition, we have 2^4 possible values for the nibble $TK_{19}[1]$. Therefore, we have $2^{56+4=60}$ possible values for the 19 tweakey nibbles that are involved in rounds 2, 15, 17, 19.

For the tweakey nibbles $TK_{16}[t]$ and $TK_{18}[f]$, the following relations hold:

$$\begin{aligned}
TK_{16}[t][0] &= \mathit{TK1}[l][0] \oplus \mathit{TK2}[l]\{0,1,2\} & \qquad TK_{18}[f][0] &= \mathit{TK1}[l][0] \oplus \mathit{TK2}[l]\{0,1,2,3\}\\
TK_{16}[t][1] &= \mathit{TK1}[l][1] \oplus \mathit{TK2}[l]\{1,2,3\} & \qquad TK_{18}[f][1] &= \mathit{TK1}[l][1] \oplus \mathit{TK2}[l]\{0,1,2\}\\
TK_{16}[t][2] &= \mathit{TK1}[l][2] \oplus \mathit{TK2}[l]\{0,2\} & \qquad TK_{18}[f][2] &= \mathit{TK1}[l][2] \oplus \mathit{TK2}[l]\{1,2,3\}\\
TK_{16}[t][3] &= \mathit{TK1}[l][3] \oplus \mathit{TK2}[l]\{1,3\} & \qquad TK_{18}[f][3] &= \mathit{TK1}[l][3] \oplus \mathit{TK2}[l]\{0,2\},
\end{aligned}$$

where $t = 0, 1, 2,\ f = 2, 0, 4$ and $l = 0, 1, 2$, respectively. From the above relations we can deduce $\mathit{TK1}[l]$, $\mathit{TK2}[l]$. Therefore, we have $2^{2\times3\times4=24}$ possible values for these 6 nibbles. Moreover, the knowledge of $\mathit{TK1}[l]$, $\mathit{TK2}[l]$ allows us to deduce the values of $ETK_1[1, 4, 5, 6, 14]$. Hence, we have 2^{24} possible values for the 10 tweakey nibbles that are involved in rounds 1, 16, 18.

For the tweakey nibbles $ETK_1[t]$ and $TK_{18}[f]$, the following relations hold:

$$\begin{aligned}
ETK_1[t][0] &= \mathit{TK1}[l][0] \oplus \mathit{TK2}[l]\{0\} & \qquad TK_{18}[f][0] &= \mathit{TK1}[l][0] \oplus \mathit{TK2}[l]\{0,1,2,3\}\\
ETK_1[t][1] &= \mathit{TK1}[l][1] \oplus \mathit{TK2}[l]\{1\} & \qquad TK_{18}[f][1] &= \mathit{TK1}[l][1] \oplus \mathit{TK2}[l]\{0,1,2\}\\
ETK_1[t][2] &= \mathit{TK1}[l][2] \oplus \mathit{TK2}[l]\{2\} & \qquad TK_{18}[f][2] &= \mathit{TK1}[l][2] \oplus \mathit{TK2}[l]\{1,2,3\}\\
ETK_1[t][3] &= \mathit{TK1}[l][3] \oplus \mathit{TK2}[l]\{3\} & \qquad TK_{18}[f][3] &= \mathit{TK1}[l][3] \oplus \mathit{TK2}[l]\{0,2\},
\end{aligned}$$

where $t = 3, 9, 11,\ f = 7, 6, 5$ and $l = 3, 4, 6$, respectively. From the above relations we can deduce $\mathit{TK1}[l]$, $\mathit{TK2}[l]$. Moreover, the knowledge of $\mathit{TK1}[6]$, $\mathit{TK2}[6]$ allows us to deduce the values of $TK_{16}[6]$ Therefore, we have $2^{2\times3\times4=24}$ possible values for these 7 nibbles. In addition, we have 2^8 possible values of $TK_{18}[1, 3]$. Hence, we have $2^{24+8=32}$ possible values for the 9 tweakey nibbles that are involved in rounds 1, 16, 18.

Table A.1: SKINNY-64-128 tweakey relations for round $i = 15, 16, \cdots, 19$ ($L_1^h = P_T^h, L_2^h = (LFSR \circ P_T)^h$)

<table>
<tr><td colspan="4">Round $i = 15$, $TK_i[j, j = 0:7] = L_1^8(\mathit{TK1}[l]) \oplus L_2^8(\mathit{TK2}[l])$, $l = 8, 9, 10, 11, 12, 13, 14, 15$
and Round $i = 16$, $TK_i[j, j = 0:7] = L_1^8(\mathit{TK1}[l]) \oplus L_2^8(\mathit{TK2}[l])$, $l = 0, 1, 2, 3, 4, 5, 6, 7$</td></tr>
<tr><td>$TK_i[j][0]$</td><td>$TK_i[j][1]$</td><td>$TK_i[j][2]$</td><td>$TK_i[j][3]$</td></tr>
<tr><td>$\mathit{TK1}[l][0] \oplus \mathit{TK2}[l]\{0,1,2\}$</td><td>$\mathit{TK1}[l][1] \oplus \mathit{TK2}[l]\{1,2,3\}$</td><td>$\mathit{TK1}[l][2] \oplus \mathit{TK2}[l]\{0,2\}$</td><td>$\mathit{TK1}[l][3] \oplus \mathit{TK2}[l]\{1,3\}$</td></tr>
<tr><td colspan="4">Round $i = 17$, $TK_i[j, j = 0:7] = L_1^9(\mathit{TK1}[l]) \oplus L_2^9(\mathit{TK2}[l])$, $l = 9, 15, 8, 13, 10, 14, 12, 11$
and Round $i = 18$, $TK_i[j, j = 0:7] = L_1^9(\mathit{TK1}[l]) \oplus L_2^9(\mathit{TK2}[l])$, $l = 1, 7, 0, 5, 2, 6, 4, 3$</td></tr>
<tr><td>$TK_i[j][0]$</td><td>$TK_i[j][1]$</td><td>$TK_i[j][2]$</td><td>$TK_i[j][3]$</td></tr>
<tr><td>$\mathit{TK1}[l][0] \oplus \mathit{TK2}[l]\{0,1,2,3\}$</td><td>$\mathit{TK1}[l][1] \oplus \mathit{TK2}[l]\{0,1,2\}$</td><td>$\mathit{TK1}[l][2] \oplus \mathit{TK2}[l]\{1,2,3\}$</td><td>$\mathit{TK1}[l][3] \oplus \mathit{TK2}[l]\{0,2\}$</td></tr>
<tr><td colspan="4">Round $i = 19$, $TK_i[j, j = 0:7] = L_1^{10}(\mathit{TK1}[l]) \oplus L_2^{10}(\mathit{TK2}[l])$, $l = 15, 11, 9, 14, 8, 12, 10, 13$</td></tr>
<tr><td>$TK_i[j][0]$</td><td>$TK_i[j][1]$</td><td>$TK_i[j][2]$</td><td>$TK_i[j][3]$</td></tr>
<tr><td>$\mathit{TK1}[l][0] \oplus \mathit{TK2}[l]\{0,1,3\}$</td><td>$\mathit{TK1}[l][1] \oplus \mathit{TK2}[l]\{0,1,2,3\}$</td><td>$\mathit{TK1}[l][2] \oplus \mathit{TK2}[l]\{0,1,2\}$</td><td>$\mathit{TK1}[l][3] \oplus \mathit{TK2}[l]\{1,2,3\}$</td></tr>
</table>

Table A.2: SKINNY-64-128 equivlant tweakey relations for round $i = 1, 2$ ($L_1^h = P_T^h, L_2^h = (LFSR \circ P_T)^h$)

<table>
<tr><td colspan="4">Round $i = 1$, $ETK_i[j, j = 0:15] = TK1[l] \oplus TK2[l], l = 0, 1, 2, 3, 0, 1, 2, 3, 7, 4, 5, 6, 0, 1, 2, 3$</td></tr>
<tr><td>$ETK_i[j][0]$</td><td>$ETK_i[j][1]$</td><td>$ETK_i[j][2]$</td><td>$ETK_i[j][3]$</td></tr>
<tr><td>$TK1[l][0] \oplus TK2[l][0]$</td><td>$TK1[l][1] \oplus TK2[l][1]$</td><td>$TK1[l][2] \oplus TK2[l][2]$</td><td>$TK1[l][3] \oplus TK2[l][3]$</td></tr>
<tr><td colspan="4">Round $i = 2$, $ETK_i[j, j = 0:15] = L_1(TK1[l]) \oplus L_2(TK2[l])$, $l = 9, 15, 8, 13, 9, 15, 8, 13, 11, 10, 14, 12, 9, 15, 8, 13$</td></tr>
<tr><td>$ETK_i[j][0]$</td><td>$ETK_i[j][1]$</td><td>$ETK_i[j][2]$</td><td>$ETK_i[j][3]$</td></tr>
<tr><td>$TK1[l][0] \oplus TK2[l]\{2, 3\}$</td><td>$TK1[l][1] \oplus TK2[l][0]$</td><td>$TK1[l][2] \oplus TK2[l][1]$</td><td>$TK1[l][3] \oplus TK2[l][2]$</td></tr>
</table>

Appendix B

SKINNY-128-256 Key Schedule Relations

Tables B.1, B.2 illustrate the tweakey and equivalent tweakey relations, respectively.

Table B.1: SKINNY-128-256 tweakey relations for round $i = 15, 16, \cdots, 19$ ($L_1^h = P_T^h, L_2^h = (LFSR \circ P_T)^h$)

Round $i = 15$, $TK_i[j, j = 0:7] = L_1^8(TK1[l]) \oplus L_2^8(TK2[l])$, $l = 8, 9, 10, 11, 12, 13, 14, 15$ and Round $i = 16$, $TK_i[j, j = 0:7] = L_1^8(TK1[l]) \oplus L_2^8(TK2[l])$, $l = 0, 1, 2, 3, 4, 5, 6, 7$			
$TK_i[j][0]$	$TK_i[j][1]$	$TK_i[j][2]$	$TK_i[j][3]$
$TK1[l][0] \oplus TK2[l]\{0, 4, 6\}$	$TK1[l][1] \oplus TK2[l]\{1, 5, 7\}$	$TK1[l][2] \oplus TK2[l]\{0, 2\}$	$TK1[l][3] \oplus TK2[l]\{1, 3\}$
$TK_i[j][4]$	$TK_i[j][5]$	$TK_i[j][6]$	$TK_i[j][7]$
$TK1[l][4] \oplus TK2[l]\{2, 4\}$	$TK1[l][5] \oplus TK2[l]\{3, 5\}$	$TK1[l][6] \oplus TK2[l]\{4, 6\}$	$TK1[l][7] \oplus TK2[l]\{5, 7\}$
Round $i = 17$, $TK_i[j, j = 0:7] = L_1^9(TK1[l]) \oplus L_2^9(TK2[l])$, $l = 9, 15, 8, 13, 10, 14, 12, 11$ and Round $i = 18$, $TK_i[j, j = 0:7] = L_1^9(TK1[l]) \oplus L_2^9(TK2[l])$, $l = 1, 7, 0, 5, 2, 6, 4, 3$			
$TK_i[j][0]$	$TK_i[j][1]$	$TK_i[j][2]$	$TK_i[j][3]$
$TK1[l][0] \oplus TK2[l]\{3, 7\}$	$TK1[l][1] \oplus TK2[l]\{0, 4, 6\}$	$TK1[l][2] \oplus TK2[l]\{1, 5, 7\}$	$TK1[l][3] \oplus TK2[l]\{0, 2\}$
$TK_i[j][4]$	$TK_i[j][5]$	$TK_i[j][6]$	$TK_i[j][7]$
$TK1[l][4] \oplus TK2[l]\{1, 3\}$	$TK1[l][5] \oplus TK2[l]\{2, 4\}$	$TK1[l][6] \oplus TK2[l]\{3, 5\}$	$TK1[l][7] \oplus TK2[l]\{4, 6\}$
Round $i = 19$, $TK_i[j, j = 0:7] = L_1^{10}(TK1[l]) \oplus L_2^{10}(TK2[l])$, $l = 15, 11, 9, 14, 8, 12, 10, 13$			
$TK_i[j][0]$	$TK_i[j][1]$	$TK_i[j][2]$	$TK_i[j][3]$
$TK1[l][0] \oplus TK2[l]\{2, 6\}$	$TK1[l][1] \oplus TK2[l]\{3, 7\}$	$TK1[l][2] \oplus TK2[l]\{0, 4, 6\}$	$TK1[l][3] \oplus TK2[l]\{1, 5, 7\}$
$TK_i[j][4]$	$TK_i[j][5]$	$TK_i[j][6]$	$TK_i[j][7]$
$TK1[l][4] \oplus TK2[l]\{0, 2\}$	$TK1[l][5] \oplus TK2[l]\{1, 3\}$	$TK1[l][6] \oplus TK2[l]\{2, 4\}$	$TK1[l][7] \oplus TK2[l]\{3, 5\}$

Table B.2: SKINNY-128-256 equivlant tweakey relations for round $i = 1, 2$ ($L_1^h = P_T^h, L_2^h = (LFSR \circ P_T)^h$)

<table>
<tr><td colspan="4">Round $i = 1$, $ETK_i[j, j = 0 : 15] = TK1[l] \oplus TK2[l], l = 0, 1, 2, 3, 0, 1, 2, 3, 7, 4, 5, 6, 0, 1, 2, 3$</td></tr>
<tr><td>$ETK_i[j][0]$</td><td>$ETK_i[j][1]$</td><td>$ETK_i[j][2]$</td><td>$ETK_i[j][3]$</td></tr>
<tr><td>$TK1[l][0] \oplus TK2[l][0]$</td><td>$TK1[l][1] \oplus TK2[l][1]$</td><td>$TK1[l][2] \oplus TK2[l][2]$</td><td>$TK1[l][3] \oplus TK2[l][3]$</td></tr>
<tr><td>$ETK_i[j][4]$</td><td>$ETK_i[j][5]$</td><td>$ETK_i[j][6]$</td><td>$ETK_i[j][7]$</td></tr>
<tr><td>$TK1[l][4] \oplus TK2[l][4]$</td><td>$TK1[l][5] \oplus TK2[l][5]$</td><td>$TK1[l][6] \oplus TK2[l][6]$</td><td>$TK1[l][7] \oplus TK2[l][7]$</td></tr>
<tr><td colspan="4">Round $i = 2$, $ETK_i[j, j = 0 : 15] = L_1(TK1[l]) \oplus L_2(TK2[l])$, $l = 9, 15, 8, 13, 9, 15, 8, 13, 11, 10, 14, 12, 9, 15, 8, 13$</td></tr>
<tr><td>$ETK_i[j][0]$</td><td>$ETK_i[j][1]$</td><td>$ETK_i[j][2]$</td><td>$ETK_i[j][3]$</td></tr>
<tr><td>$TK1[l][0] \oplus TK2[l]\{5, 7\}$</td><td>$TK1[l][1] \oplus TK2[l][0]$</td><td>$TK1[l][2] \oplus TK2[l][1]$</td><td>$TK1[l][3] \oplus TK2[l][2]$</td></tr>
<tr><td>$ETK_i[j][4]$</td><td>$ETK_i[j][5]$</td><td>$ETK_i[j][6]$</td><td>$ETK_i[j][7]$</td></tr>
<tr><td>$TK1[l][4] \oplus TK2[l][3]$</td><td>$TK1[l][5] \oplus TK2[l][4]$</td><td>$TK1[l][6] \oplus TK2[l][5]$</td><td>$TK1[l][7] \oplus TK2[l][6]$</td></tr>
</table>

Appendix C

SKINNY-64-64/SKINNY-128-128 Key Schedule Relations

Tables C.1, C.2 illustrate the tweakey and equivalent tweakey relations, respectively.

Table C.1: SKINNY-64-64 and SKINNY-128-128 tweakey relations for round $i = 15, 16, 17$

Round $i = 15$	$TK_i[j, j = 0:7] = TK1[l]$, $l = 8, 9, 10, 11, 12, 13, 14, 15$
Round $i = 16$	$TK_i[j, j = 0:7] = TK1[l]$, $l = 0, 1, 2, 3, 4, 5, 6, 7$
Round $i = 17$	$TK_i[j, j = 0:7] = TK1[l]$, $l = 9, 15, 8, 13, 10, 14, 12, 11$

Table C.2: SKINNY-64-64 and SKINNY-128-128 equivlant tweakey relations for round $i = 1, 2$

Round $i = 1$	$ETK_i[j, j = 0:15] = TK1[l], l = 0, 1, 2, 3, 0, 1, 2, 3{,}7, 4, 5, 6, 0, 1, 2, 3$
Round $i = 2$	$ETK_i[j, j = 0:15] = TK1[l], l = 9, 15, 8, 13, 9, 15, 8, 13{,}11, 10, 14, 12, 9, 15, 8, 13$

Appendix D

SKINNY-64-192/SKINNY-128-384 Key Schedule Relations

Tables D.1, D.2 (resp. D.3, D.4) illustrate the tweakey and equivalent tweakey relations of SKINNY-64-192 (resp. SKINNY-128-384).

Table D.1: SKINNY-64-192 tweakey relations for round $i = 15, 16, \cdots, 21$ ($L_1^h = P_T^h, L_2^h = (LFSR \circ P_T)^h$)

<table>
<tr><td colspan="4">Round $i = 15$, $TK_i[j, j = 0:7] = L_1^8(TK1[l]) \oplus L_2^8(TK2[l]) \oplus L_2^8(TK3[l])$, $l = 8, 9, 10, 11, 12, 13, 14, 15$
and Round $i = 16$, $TK_i[j, j = 0:7] = L_1^8(TK1[l]) \oplus L_2^8(TK2[l]) \oplus L_2^8(TK3[l])$, $l = 0, 1, 2, 3, 4, 5, 6, 7$</td></tr>
<tr><td>$TK_i[j][0]$</td><td>$TK_i[j][1]$</td><td>$TK_i[j][2]$</td><td>$TK_i[j][3]$</td></tr>
<tr><td>$TK1[l][0] \oplus TK2[l]\{0,1,2\}$ $\oplus TK3[l]\{1,2,3\}$</td><td>$TK1[l][1] \oplus TK2[l]\{1,2,3\}$ $\oplus TK3[l]\{0,2\}$</td><td>$TK1[l][2] \oplus TK2[l]\{0,2\}$ $\oplus TK3[l]\{1,3\}$</td><td>$TK1[l][3] \oplus TK2[l]\{1,3\}$ $\oplus TK3[l]\{0,2,3\}$</td></tr>
<tr><td colspan="4">Round $i = 17$, $TK_i[j, j = 0:7] = L_1^9(TK1[l]) \oplus L_2^9(TK2[l]) \oplus L_2^9(TK3[l])$, $l = 9, 15, 8, 13, 10, 14, 12, 11$
and Round $i = 18$, $TK_i[j, j = 0:7] = L_1^9(TK1[l]) \oplus L_2^9(TK2[l]) \oplus L_2^9(TK3[l])$, $l = 1, 7, 0, 5, 2, 6, 4, 3$</td></tr>
<tr><td>$TK_i[j][0]$</td><td>$TK_i[j][1]$</td><td>$TK_i[j][2]$</td><td>$TK_i[j][3]$</td></tr>
<tr><td>$TK1[l][0] \oplus TK2[l]\{0,1,2,3\}$ $\oplus TK3[l]\{0,2\}$</td><td>$TK1[l][1] \oplus TK2[l]\{0,1,2\}$ $\oplus TK3[l]\{1,3\}$</td><td>$TK1[l][2] \oplus TK2[l]\{1,2,3\}$ $\oplus TK3[l]\{0,2,3\}$</td><td>$TK1[l][3] \oplus TK2[l]\{0,2\}$ $\oplus TK3[l]\{0,1\}$</td></tr>
<tr><td colspan="4">Round $i = 19$, $TK_i[j, j = 0:7] = L_1^{10}(TK1[l]) \oplus L_2^{10}(TK2[l]) \oplus L_2^{10}(TK3[l])$, $l = 15, 11, 9, 14, 8, 12, 10, 13$
and Round $i = 20$, $TK_i[j, j = 0:7] = L_1^{10}(TK1[l]) \oplus L_2^{10}(TK2[l]) \oplus L_2^{10}(TK3[l])$, $l = 7, 3, 1, 6, 0, 4, 2, 5$</td></tr>
<tr><td>$TK_i[j][0]$</td><td>$TK_i[j][1]$</td><td>$TK_i[j][2]$</td><td>$TK_i[j][3]$</td></tr>
<tr><td>$TK1[l][0] \oplus TK2[l]\{0,1,3\}$ $\oplus TK3[l]\{1,3\}$</td><td>$TK1[l][1] \oplus TK2[l]\{0,1,2,3\}$ $\oplus TK3[l]\{0,2,3\}$</td><td>$TK1[l][2] \oplus TK2[l]\{0,1,2\}$ $\oplus TK3[l]\{0,1\}$</td><td>$TK1[l][3] \oplus TK2[l]\{1,2,3\}$ $\oplus TK3[l]\{1,2\}$</td></tr>
<tr><td colspan="4">Round $i = 21$, $TK_i[j, j = 0:7] = L_1^{11}(TK1[l]) \oplus L_2^{11}(TK2[l]) \oplus L_2^{11}(TK3[l])$, $l = 11, 13, 15, 12, 9, 10, 8, 14$</td></tr>
<tr><td>$TK_i[j][0]$</td><td>$TK_i[j][1]$</td><td>$TK_i[j][2]$</td><td>$TK_i[j][3]$</td></tr>
<tr><td>$TK1[l][0] \oplus TK2[l]\{0,3\}$ $\oplus TK3[l]\{0,2,3\}$</td><td>$TK1[l][1] \oplus TK2[l]\{0,1,3\}$ $\oplus TK3[l]\{0,1\}$</td><td>$TK1[l][2] \oplus TK2[l]\{0,1,2,3\}$ $\oplus TK3[l]\{1,2\}$</td><td>$TK1[l][3] \oplus TK2[l]\{0,1,2\}$ $\oplus TK3[l]\{2,3\}$</td></tr>
</table>

Table D.2: SKINNY-64-192 equivlant tweakey relations for round $i = 1, 2$ ($L_1^h = P_T^h, L_2^h = (LFSR \circ P_T)^h$)

Round $i = 1$, $ETK_i[j, j = 0:15] = TK1[l] \oplus TK2[l] \oplus TK3[l], l = 0, 1, 2, 3, 0, 1, 2, 3, 7, 4, 5, 6, 0, 1, 2, 3$			
$ETK_i[j][0]$	$ETK_i[j][1]$	$ETK_i[j][2]$	$ETK_i[j][3]$
$TK1[l][0] \oplus TK2[l][0]$ $\oplus TK3[l][0]$	$TK1[l][1] \oplus TK2[l][1]$ $\oplus TK3[l][1]$	$TK1[l][2] \oplus TK2[l][2]$ $\oplus TK3[l][2]$	$TK1[l][3] \oplus TK2[l][3]$ $\oplus TK3[l][3]$
Round $i = 2$, $ETK_i[j, j = 0:15] = L_1(TK1[l]) \oplus L_2(TK2[l]) \oplus L_2(TK3[l])$, $l = 9, 15, 8, 13, 9, 15, 8, 13, 11, 10, 14, 12, 9, 15, 8, 13$			
$ETK_i[j][0]$	$ETK_i[j][1]$	$ETK_i[j][2]$	$ETK_i[j][3]$
$TK1[l][0] \oplus TK2[l]\{2, 3\}$ $\oplus TK3[l]\{1\}$	$TK1[l][1] \oplus TK2[l][0]$ $\oplus TK3[l]\{2\}$	$TK1[l][2] \oplus TK2[l][1]$ $\oplus TK3[l]\{3\}$	$TK1[l][3] \oplus TK2[l][2]$ $\oplus TK3[l]\{0, 3\}$

Table D.3: SKINNY-128-384 tweakey relations for round $i = 15, 16, \cdots, 21$ ($L_1^h = P_T^h, L_2^h = (LFSR \circ P_T)^h$)

Round $i = 15$, $TK_i[j, j = 0:7] = L_1^8(TK1[l]) \oplus L_2^8(TK2[l]) \oplus L_2^8(TK3[l])$, $l = 8, 9, 10, 11, 12, 13, 14, 15$ and Round $i = 16$, $TK_i[j, j = 0:7] = L_1^8(TK1[l]) \oplus L_2^8(TK2[l]) \oplus L_2^8(TK3[l])$, $l = 0, 1, 2, 3, 4, 5, 6, 7$			
$TK_i[j][0]$	$TK_i[j][1]$	$TK_i[j][2]$	$TK_i[j][3]$
$TK1[l][0] \oplus TK2[l]\{0, 4, 6\}$ $TK3[l]\{0, 6\}$	$TK1[l][1] \oplus TK2[l]\{1, 5, 7\}$ $TK3[l]\{1, 7\}$	$TK1[l][2] \oplus TK2[l]\{0, 2\}$ $TK3[l]\{0, 2, 6\}$	$TK1[l][3] \oplus TK2[l]\{1, 3\}$ $TK3[l]\{1, 3, 7\}$
$TK_i[j][4]$	$TK_i[j][5]$	$TK_i[j][6]$	$TK_i[j][7]$
$TK1[l][4] \oplus TK2[l]\{2, 4\}$ $TK3[l]\{0, 2, 4, 6\}$	$TK1[l][5] \oplus TK2[l]\{3, 5\}$ $TK3[l]\{1, 3, 5, 7\}$	$TK1[l][6] \oplus TK2[l]\{4, 6\}$ $TK3[l]\{0, 2, 4\}$	$TK1[l][7] \oplus TK2[l]\{5, 7\}$ $TK3[l]\{1, 3, 5\}$
Round $i = 17$, $TK_i[j, j = 0:7] = L_1^9(TK1[l]) \oplus L_2^9(TK2[l]) \oplus L_2^9(TK3[l])$, $l = 9, 15, 8, 13, 10, 14, 12, 11$ and Round $i = 18$, $TK_i[j, j = 0:7] = L_1^9(TK1[l]) \oplus L_2^9(TK2[l]) \oplus L_2^9(TK3[l])$, $l = 1, 7, 0, 5, 2, 6, 4, 3$			
$TK_i[j][0]$	$TK_i[j][1]$	$TK_i[j][2]$	$TK_i[j][3]$
$TK1[l][0] \oplus TK2[l]\{3, 7\}$ $TK3[l]\{1, 7\}$	$TK1[l][1] \oplus TK2[l]\{0, 4, 6\}$ $TK3[l]\{0, 2, 6\}$	$TK1[l][2] \oplus TK2[l]\{1, 5, 7\}$ $TK3[l]\{1, 3, 7\}$	$TK1[l][3] \oplus TK2[l]\{0, 2\}$ $TK3[l]\{0, 2, 4, 6\}$
$TK_i[j][4]$	$TK_i[j][5]$	$TK_i[j][6]$	$TK_i[j][7]$
$TK1[l][4] \oplus TK2[l]\{1, 3\}$ $TK3[l]\{1, 3, 5, 7\}$	$TK1[l][5] \oplus TK2[l]\{2, 4\}$ $TK3[l]\{0, 2, 4\}$	$TK1[l][6] \oplus TK2[l]\{3, 5\}$ $TK3[l]\{1, 3, 5\}$	$TK1[l][7] \oplus TK2[l]\{4, 6\}$ $TK3[l]\{2, 4, 6\}$
Round $i = 19$, $TK_i[j, j = 0:7] = L_1^{10}(TK1[l]) \oplus L_2^{10}(TK2[l]) \oplus L_2^{10}(TK3[l])$, $l = 15, 11, 9, 14, 8, 12, 10, 13$ and Round $i = 20$, $TK_i[j, j = 0:7] = L_1^{10}(TK1[l]) \oplus L_2^{10}(TK2[l]) \oplus L_2^{10}(TK3[l])$, $l = 7, 3, 1, 6, 0, 4, 2, 5$			
$TK_i[j][0]$	$TK_i[j][1]$	$TK_i[j][2]$	$TK_i[j][3]$
$TK1[l][0] \oplus TK2[l]\{2, 6\}$ $TK3[l]\{0, 2, 6\}$	$TK1[l][1] \oplus TK2[l]\{3, 7\}$ $TK3[l]\{1, 3, 7\}$	$TK1[l][2] \oplus TK2[l]\{0, 4, 6\}$ $TK3[l]\{0, 2, 4, 6\}$	$TK1[l][3] \oplus TK2[l]\{1, 5, 7\}$ $TK3[l]\{1, 3, 5, 7\}$
$TK_i[j][4]$	$TK_i[j][5]$	$TK_i[j][6]$	$TK_i[j][7]$
$TK1[l][4] \oplus TK2[l]\{0, 2\}$ $TK3[l]\{0, 2, 4\}$	$TK1[l][5] \oplus TK2[l]\{1, 3\}$ $TK3[l]\{1, 3, 5\}$	$TK1[l][6] \oplus TK2[l]\{2, 4\}$ $TK3[l]\{2, 4, 6\}$	$TK1[l][7] \oplus TK2[l]\{3, 5\}$ $TK3[l]\{3, 5, 7\}$
Round $i = 21$, $TK_i[j, j = 0:7] = L_1^{11}(TK1[l]) \oplus L_2^{11}(TK2[l]) \oplus L_2^{11}(TK3[l])$, $l = 11, 13, 15, 12, 9, 10, 8, 14$			
$TK_i[j][0]$	$TK_i[j][1]$	$TK_i[j][2]$	$TK_i[j][3]$
$TK1[l][0] \oplus TK2[l]\{1, 5\}$ $TK3[l]\{1, 3, 7\}$	$TK1[l][1] \oplus TK2[l]\{2, 6\}$ $TK3[l]\{0, 2, 4, 6\}$	$TK1[l][2] \oplus TK2[l]\{3, 7\}$ $TK3[l]\{1, 3, 5, 7\}$	$TK1[l][3] \oplus TK2[l]\{0, 4, 6\}$ $TK3[l]\{0, 2, 4\}$
$TK_i[j][4]$	$TK_i[j][5]$	$TK_i[j][6]$	$TK_i[j][7]$
$TK1[l][4] \oplus TK2[l]\{1, 5, 7\}$ $TK3[l]\{1, 3, 5\}$	$TK1[l][5] \oplus TK2[l]\{0, 2\}$ $TK3[l]\{2, 4, 6\}$	$TK1[l][6] \oplus TK2[l]\{1, 3\}$ $TK3[l]\{3, 5, 7\}$	$TK1[l][7] \oplus TK2[l]\{2, 4\}$ $TK3[l]\{0, 4\}$

Table D.4: SKINNY-128-384 equivlant tweakey relations for round $i = 1, 2$ ($L_1^h = P_T^h, L_2^h = (LFSR \circ P_T)^h$)

Round $i = 1$, $ETK_i[j, j = 0:15] = TK1[l] \oplus TK2[l] \oplus TK3[l]$,$l = 0, 1, 2, 3, 0, 1, 2, 3$,$7, 4, 5, 6, 0, 1, 2, 3$			
$ETK_i[j][0]$	$ETK_i[j][1]$	$ETK_i[j][2]$	$ETK_i[j][3]$
$TK1[l][0] \oplus TK2[l][0]$ $\oplus TK3[l][0]$	$TK1[l][1] \oplus TK2[l][1]$ $\oplus TK3[l][1]$	$TK1[l][2] \oplus TK2[l][2]$ $\oplus TK3[l][2]$	$TK1[l][3] \oplus TK2[l][3]$ $\oplus TK3[l][3]$
$ETK_i[j][4]$	$ETK_i[j][5]$	$ETK_i[j][6]$	$ETK_i[j][7]$
$TK1[l][4] \oplus TK2[l][4]$ $\oplus TK3[l][4]$	$TK1[l][5] \oplus TK2[l][5]$ $\oplus TK3[l][5]$	$TK1[l][6] \oplus TK2[l][6]$ $\oplus TK3[l][6]$	$TK1[l][7] \oplus TK2[l][7]$ $\oplus TK3[l][7]$
Round $i = 2$, $ETK_i[j, j = 0:15] = L_1(TK1[l]) \oplus L_2(TK2[l]) \oplus L_2(TK3[l])$, $l = 9, 15, 8, 13, 9, 15, 8, 13, 11, 10, 14, 12, 9, 15, 8, 13$			
$ETK_i[j][0]$	$ETK_i[j][1]$	$ETK_i[j][2]$	$ETK_i[j][3]$
$TK1[l][0] \oplus TK2[l]\{5, 7\}$ $\oplus TK3[l][1]$	$TK1[l][1] \oplus TK2[l][0]$ $\oplus TK3[l][2]$	$TK1[l][2] \oplus TK2[l][1]$ $\oplus TK3[l][3]$	$TK1[l][3] \oplus TK2[l][2]$ $\oplus TK3[l][4]$
$ETK_i[j][4]$	$ETK_i[j][5]$	$ETK_i[j][6]$	$ETK_i[j][7]$
$TK1[l][4] \oplus TK2[l][3]$ $\oplus TK3[l][5]$	$TK1[l][5] \oplus TK2[l][4]$ $\oplus TK3[l][6]$	$TK1[l][6] \oplus TK2[l][5]$ $\oplus TK3[l][7]$	$TK1[l][7] \oplus TK2[l][6]$ $\oplus TK3[l]\{0, 6\}$

Appendix E

Chosen Plaintext MitM Attack on 5-round Kuznyechik

The authors in [14] implied that their attack can only work in the chosen ciphertext model. In this appendix, we show how we can tweak their attack to work in the plaintext model. Figure E.1 illustrates our 3-round distinguisher which starts at x_1 and ends at y_4. The δ-set is chosen at byte 15 and the multiset is computed at byte 15. Our distinguisher is based on the following proposition:

Proposition 11 *If a message m belongs to a pair of states conforming to the truncated differential characteristic of Figure E.1, then the multiset of differences $\Delta y_4[15]$ obtained from the δ-set constructed from m in $x_1[15]$ is fully determined by the following 19 bytes: $\Delta x_1[15], x_2, y_4[15]$ and $\Delta y_4[15]$.*

Proposition 11 can be proved using the same approach used to prove Proposition 10.

Offline Phase. In this phase, we compute the multiset at $y_4[15]$ using the 19 byte parameters mentioned in Proposition 11, i.e., we have $2^{19\times8} = 2^{152}$ multiset out of $2^{467.6}$ theoretically possible ones.

Data Collection. The probability of the truncated differential characteristic can be evaluated as follows: transition from z_4 to y_4 over L^{-1} $(16 \rightarrow 1)$ of probability $2^{-15\times8} = 2^{-120}$. Therefore, we need to collect 2^{120} message pairs to guarantee that there exist one message pair which conform to the truncated path. We use the same structure that is used in the 6-round attack. Hence, we need to query 2^{113} chosen plaintext.

Key Recovery. In order to build the δ-set and compute the multiset, we need to guess K_6. The key suggestions for the 16 bytes K_6 can be obtained by guessing $\Delta y_4[15]$. Therefore, we have 2^8 values for the 16 bytes key K_6.

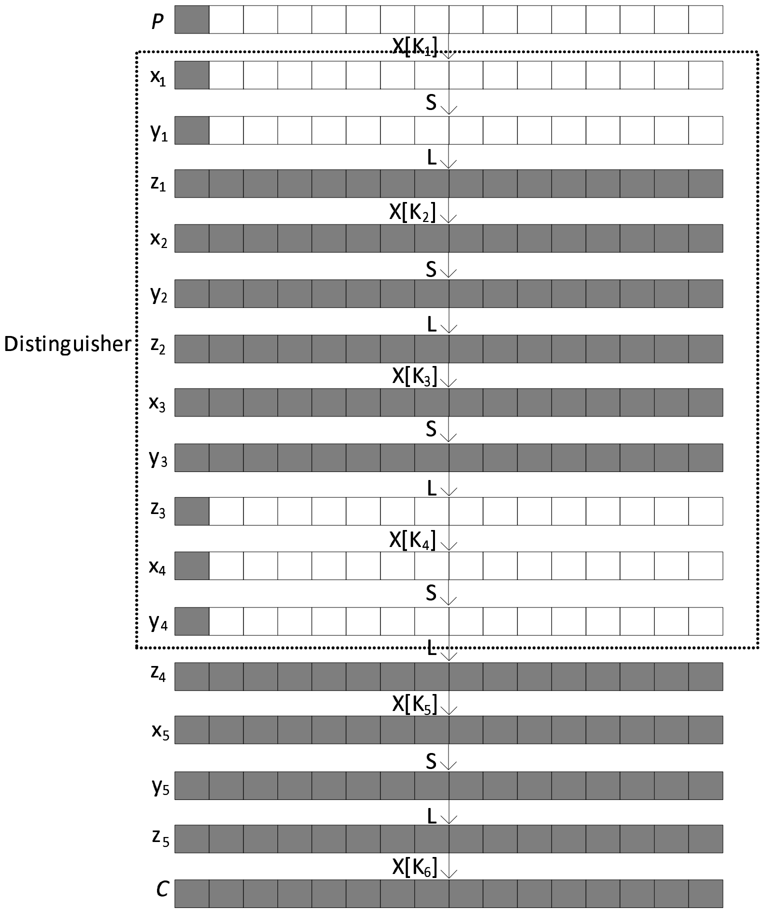


Figure E.1: Kuznyechik 5-round attack

The probability of finding a match in the table with the wrong key is $2^{152}/2^{467.6} = 2^{-315.6}$. Therefore, the number of key candidates of K_6 after launching the attack is $2^{120+8-315.6} = 2^{-187.6}$, i.e., our attack will find one right value for K_6. Then, the master key can be recovered by guessing K_5 using two plaintext/ciphertext pairs.

Attack complexity. The memory complexity is $2^{152} \times 512/128 = 2^{154}$ 128-bit. The data complexity is 2^{113} chosen plaintext. The time complexity is $2^{19\times 8} \times 2^8 \times 3/5 + 2^{120} \times 2^8 \times 2^8 \times 2/5 + 2 \times 2^{128} \approx 2^{159.3} + 2^{134.7} + 2^{129} \approx 2^{159.3}$ encryptions. Our attack has an online time complexity of $2^{134.7}$ encryptions. Therefore, this attack reduces the online time complexity of the previous attack [14] by a factor of $2^{5.6}$ with the same data and a non significant increase in the memory complexity.

Appendix F

Key Schedule Relations for SPARX-128/128

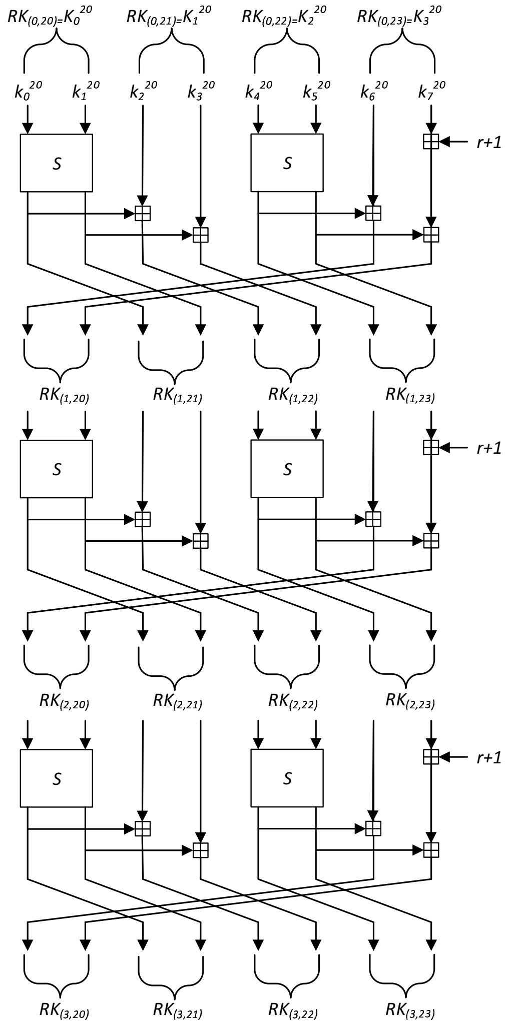


Figure F.1: Key secluded relations of SPARX-128/128

Appendix G

Zero-Correlation Distinguisher for SPARX-64/128

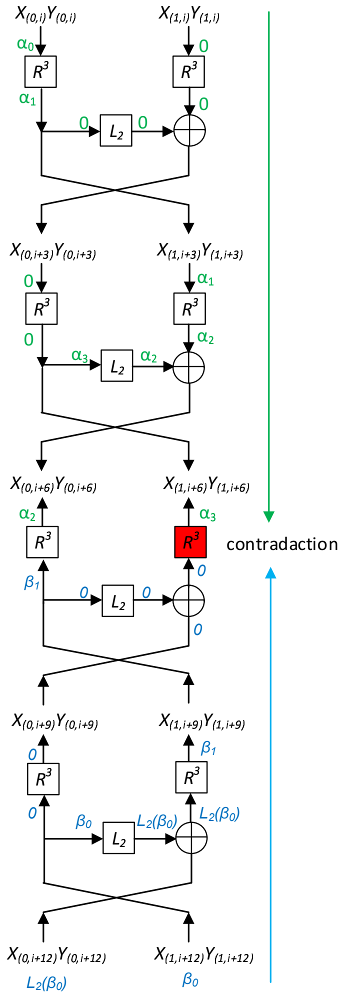


Figure G.1: A 12-round zero-correlation distinguisher of SPARX-64/128, where α_i, β_j are 32-bit non-zero linear masks and $\mathbf{0}$ denotes $0x0000\ 0x0000$ linear mask