Differential analysis of the symmetric key cryptosystem

Tjuawinata, Ivan

2018

Tjuawinata, I. (2018). Differential analysis of the symmetric key cryptosystem. Doctoral thesis, Nanyang Technological University, Singapore.

<http://hdl.handle.net/10356/73150>

<https://doi.org/10.32657/10356/73150>

NANYANG
TECHNOLOGICAL
UNIVERSITY

DIFFERENTIAL ANALYSIS OF THE SYMMETRIC KEY CRYPTOSYSTEM

IVAN TJUAWINATA

SCHOOL OF PHYSICAL AND MATHEMATICAL SCIENCES

2017

DIFFERENTIAL ANALYSIS OF THE SYMMETRIC KEY CRYPTOSYSTEM

IVAN TJUAWINATA

SCHOOL OF PHYSICAL AND MATHEMATICAL SCIENCES

A thesis submitted to Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2017

Acknowledgments

This thesis would not have been possible without the guidance and support of a great many people.

First and foremost, I would like to express my gratitude to my supervisor, professor Wu Hongjun, for his guidance, advice, encouragement and support during my PhD years. I am deeply indebted to him for his understanding and kindness.

I would also like to express my gratitude to the reviewers for reviewing and giving advices to improve this thesis.

I want to thank the professors and lecturers in NTU for providing me with the knowledge that is essential for writing this thesis and pursuing my goals.

I would like to give a special thank to my senior Huang Tao for his insightful suggestions, his instructive explanations and comments.

I want to thank my friends for helpful discussions and cooperations. I would also like to thank them for their time and their willingness to share their knowledge and experience.

Last but not least, I want to gratefully thank my father, my mother and my sister for their great love, support, encouragement, and inspiration.

June 19, 2017

Contents

Acknowledgments	3
Contents	5
Abstract	11
1 Introduction	13
1.1 Symmetric Key Cryptography	13
1.1.1 Symmetric Key Encryption	15
1.1.1.1 Block Cipher	17
1.1.1.2 Stream Cipher	25
1.1.2 Hash Function	27
1.1.2.1 Iterated Hash Function	27
1.1.3 Authentication Encryption Scheme	29
1.2 Cryptanalysis of Symmetric Key Cryptosystem	31
1.2.1 Linear Cryptanalysis	33
1.2.2 Differential Cryptanalysis	35
1.2.2.1 Truncated Differential Cryptanalysis	36
1.2.2.2 Impossible Differential Cryptanalysis	38
1.2.2.3 Boomerang Attack	38
1.2.3 Differential-Linear Cryptanalysis	40
1.2.4 Security Analysis Related to Padding	41

1.3	Thesis Outline	42
2	Cryptanalysis of Simpira v2	45
2.1	Introduction	45
2.2	Preliminaries	48
2.2.1	Notation	48
2.2.2	Description of Simpira	48
2.2.3	Even-Mansour Construction	49
2.3	Truncated Differential Attack on Simpira-3	50
2.3.1	Distinguishing Attacks on Simpira-3	52
2.3.1.1	8-Round Distinguishing Attack	52
2.3.1.2	9-Round Distinguishing Attack	54
2.3.1.3	9-Round Key Recovery Using Truncated Differential	56
2.4	Impossible Differential Attack on Simpira-3	57
2.4.1	Key Recovery Attack on 10-Round Simpira-3	59
2.4.2	Key Recovery Attack on 9-Round Simpira-3	60
2.5	Boomerang Attack on Simpira-3	62
2.5.1	Distinguishing Attack on 10- Rounds Simpira-3	63
2.5.2	Distinguishing Attack on 9- Rounds Simpira-3	65
2.5.3	10-Round Key Recovery Attack Using Boomerang	66
2.6	Summary of Findings	68
2.7	Conclusion	68
3	Improved Differential Cryptanalysis on Generalized Feistel Schemes	71
3.1	Introduction	71
3.1.1	Background	71
3.1.2	Our Contribution	72
3.1.3	Organization	73
3.2	Preliminaries	74

3.2.1	Generalized Feistel Schemes	74
3.2.2	Random Variable	74
3.3	Overview of [105]	75
3.4	Attack Overview	76
3.5	Type-1 Feistel Scheme	78
3.5.1	Analysis of the Type-1 Feistel Scheme	79
3.5.2	Calculation of Expected Value and Variance of $\mathcal{N}_{(\mathcal{F}, \text{CCA})}$ where $\mathcal{F}$ is an ak -round Type-1 Feistel Scheme	83
3.5.2.1	$ \{i, j, r, s\} = 4$	86
3.5.2.2	$ \{i, j, r, s\} = 3$	88
3.5.3	Comparison with Existing Result from [105]	89
3.6	Type-3 Feistel Scheme	91
3.6.1	Analysis of the Type-3 Feistel Scheme	91
3.6.2	Comparison with Existing Result from [105]	93
3.7	Alternating Feistel Scheme	94
3.7.1	Analysis of Alternating Feistel Scheme	94
3.7.1.1	Even Number of Rounds	94
3.7.1.2	Odd Number of Rounds	95
3.7.2	Comparison with Existing Result from [105]	96
3.8	Unbalanced Feistel Scheme	97
3.8.1	Analysis of Unbalanced Feistel Scheme	97
3.8.1.1	Analysis of $\text{UFN}(k', k)$ when k' divides k	98
3.8.2	Result of Distinguishability of $\text{UFN}(k', k)$ when $(k - k')$ divides k	101
3.9	Conclusion	105
4	Differential-Linear Cryptanalysis of ICEPOLE	107
4.1	Introduction	107
4.2	The ICEPOLE Authenticated Cipher	109
4.2.1	Notations	110

4.2.2	The ICEPOLE permutation P	110
4.2.3	Initialization	112
4.2.4	Processing associated data and plaintext	113
4.2.5	Tag generation	115
4.2.6	Security goals of ICEPOLE	115
4.3	Differential-Linear Distinguishing Attack on ICEPOLE	115
4.3.1	Constructing the differential characteristics	117
4.3.2	Constructing the linear characteristic	118
4.3.3	Observations to improve the attack	120
4.3.4	Concatenating the differential and linear characteristics	122
4.4	State Recovery Attack on ICEPOLE	124
4.4.1	State recovery attacks on ICEPOLE-128 and ICEPOLE-128a	124
4.4.1.1	Recovering U_0 and U_3	124
4.4.1.2	Recovering U_2	126
4.4.1.3	Recovering U_1	128
4.4.1.4	Correcting the recovered state	130
4.4.1.5	Summary of the attack	130
4.4.2	State recovery attack on ICEPOLE-256a	131
4.4.3	Implications of the state recovery attacks	133
4.5	Experimental Results	134
4.6	Conclusion	135

5 Cryptanalysis of the Authenticated Encryption Algorithm COFFE 137

5.1	Introduction	137
5.2	The COFFE Authenticated Cipher	138
5.2.1	Notations	139
5.2.2	Associated Data Processing	141
5.2.3	Initialization	141
5.2.4	Processing plaintext	142

5.2.5	Tag generation	142
5.2.6	Security goals of COFFE	143
5.3	Analysis on the COFFE scheme	144
5.3.1	Modification of β	145
5.3.2	Modification of α	151
5.4	Distinguishing Attack	152
5.5	Ciphertext Forgery Attack	154
5.5.1	Forgery Attack with Constant Message Block Size	154
5.5.2	Forgery Attack with Dynamic Message Block Size	156
5.5.3	Combination of the Existing Attacks	157
5.6	Related Key Recovery Attack	157
5.7	Conclusion	158
5.7.1	Attack Summary	158
5.7.2	Lesson Learned	159
6	Conclusion	161
	List of Publications	165
	Bibliography	167
	List of Figures	185
A	Appendix	187
A.1	Session Key Generation Analysis	187

Abstract

Differential cryptanalysis is one of the most basic and powerful attacks against symmetric key cryptosystems. The idea is to consider the effect of the difference in two inputs to the difference of the output of some cryptosystem. When the probability is high enough, this can be used to recover some information about the secret key or state value that should not be available to the attacker. Differential cryptanalysis has also been extended and improved to many different variants, some of them are truncated differential cryptanalysis, impossible differential cryptanalysis, boomerang attack and differential-linear cryptanalysis.

In this thesis, we consider the security of some symmetric key cryptosystems using the analyses of the relation of pairs of inputs and its corresponding outputs.

We first give a brief review on different symmetric key cryptosystems and some cryptanalysis technique that is mainly focused on differential cryptanalysis.

We then analyse the security of three concrete designs and one general design. We first consider the security of Simpira v2, a Feistel cipher with 3 branches. We apply truncated differential analysis, impossible differential cryptanalysis and boomerang attack to study the number of rounds that can be attacked using these analysis. We exploit the asymmetry of encryption and decryption of the three branches Feistel Design and analyze the decryption function of Simpira v2. Based on the three variants of the differential attack, we found an attack with 2^{24} and 2^{70} queries for partial key recovery attack in the 9- and 10- round Simpira-3 v2.

Next, we analyse five types of Generalized Feistel Schemes with ideal round function. In this chapter, we provide a lower bound for the maximum number of rounds that still can be distinguishable from a random permutation. Since the round function is assumed to be any ideal function, the attack is applicable to any block cipher design using the Generalized Feistel structure.

After that, two authenticated encryption schemes are analyzed. First, we apply differential-linear cryptanalysis on the authenticated encryption ICEPOLE. In the

attack we consider the design under the assumption that the nonce can be reused. We found that in this assumption, it is possible to recover the state value of the ICEPOLE. Furthermore, in some of the variants that do not use secret message number, the secret key can be recovered.

Lastly, we consider the hash function-based authenticated encryption COFFE. In this analysis, we assume that the underlying hash function is any ideal hash function. We are considering the possibility of a pair of distinct input can be used to recover more information about the scheme. In our case, with some specific parameter value, it is possible to distinguish a full COFFE implementation from a random permutation and in the related key setting, it is even possible to launch a key recovery attack based on it. In some pairs of parameters, this can even be done using 2 queries.

Chapter 1

Introduction

1.1 Symmetric Key Cryptography

Cryptography is a study of secure message transmission. The assumption is that we have a sender, say Alice, who wants to send a message to a recipient, say Bob. The objective is that the message should be secure from other parties, say Eve. There are two main categories of cryptography, namely symmetric key and public key cryptosystem. A symmetric key cryptosystem is based on the assumption of the existence of a secret shared by the sender and receiver that can be used to securely send messages between them. On the other hand, a public key cryptosystem uses a one-way function with a trap door to construct a secure message transmission. Although the one-way function can be public, the owner of the function keeps the trap door secret for him to “invert” the function. In this thesis, we focus on symmetric key cryptosystem. The security of the message transmission mainly involves four factors [146]:

1. Confidentiality: The original message should not be revealed to Eve. For this, the cryptosystem involved is called the encryption system. An encryption system is a pair of keyed functions E_K , called the encryption scheme and $D_{K'}$, called the decryption scheme. The keys used in the encryption and decryption

scheme are called the encryption and decryption key respectively. When the decryption key is the appropriate key corresponding to the encryption key, for any message M , we should have $D_{K'}(E_K(M)) = M$.

Furthermore, using a secure encryption scheme, without knowing the decryption key K' that corresponds to the encryption key K , the complexity for Eve to retrieve M given $E_K(M)$ should be too high to be feasible. In a symmetric key cryptosystem, the decryption key K' needs to be derivable from K . Since the decryption key can be derived from the encryption key, to ensure confidentiality, the encryption key cannot be public.

2. Integrity: Upon receiving a message M from Alice, Bob needs to be sure that the message M received is indeed the message Alice sent. The cryptosystem that achieves this factor is the hash function. A hash function is a function to construct a short “fingerprint”, or message digest, of the message being sent so Bob can recompute the function to check if the digest received from Alice along with the message matches the one he computed [128]. If it is not the same, Bob can be sure that the message has been modified while if it matches, he can have a certain level of confidence that it has not been changed along transmission.

Some examples of well known and widely used hash functions are MD5 [119], SHA-1 [8], SHA-2 [3] and Keccak [5], which has recently been selected as SHA-3 standard.

3. Authenticity: Given a message M from Alice, Bob needs to be sure that the sender is indeed Alice and not Eve. Message authentication code, or MAC, is the cryptosystem that is designed to achieve this objective. Similar to Hash functions, a MAC has an arbitrary input length and a fixed output length. The requirement of the sender authenticity makes the use of an unkeyed hash function not useful for this factor. This is because if the MAC does not contain something only the sender is supposed to have, anyone can reconstruct the

message and its MAC. Due to its similarity, it is easy to see that a MAC can be constructed by including a secret key shared by Alice and Bob in the input of a hash function. A MAC constructed using this method is called hash-based MAC. Some examples of hash-based functions are HMAC [15] and UMAC [36]. There are other methods to construct a MAC through a symmetric key cryptosystem, some of them are EMAC [37] and PMAC [39].

4. Non-repudiation: After sending message M from Alice, Bob should be certain that Alice cannot deny later that she is the one who sends the message. However, in a symmetric key setting, this cannot be done considering that the recipient also has the same secret key and hence a third party cannot distinguish whether the sender is Alice or Bob. In public key cryptosystem, this can be solved by the use of digital signature.

Having the main objectives of the study of cryptography, we review in more details some of cryptosystems that will be discussed in this thesis, namely symmetric key encryption schemes, hash functions and authenticated encryption schemes.

1.1.1 Symmetric Key Encryption

Firstly, we formally define the symmetric key encryption scheme.

Definition 1.1.1. Let $\mathcal{P}$ be the space of all possible plaintexts, $\mathcal{K}$ the space of all possible secret keys and $\mathcal{C}$ the space of all possible ciphertexts. Let $\mathcal{A}$ be the set of all possible values of other parameters that is possibly used in the encryption and decryption scheme such as nonce and parameter lengths. A symmetric key encryption scheme with the secret key $K \in \mathcal{K}$ and other parameters $A \in \mathcal{A}$ is a pair of functions $(E_{K,A} : \mathcal{P} \rightarrow \mathcal{C}, D_{K,A} : \mathcal{C} \rightarrow \mathcal{P})$ such that for any $M \in \mathcal{P}$, $D_K(E_K(M, A), A) = M$. Any intermediate value that M (or C) is changed to throughout the encryption (or decryption) is called the intermediate state value.

Here we are assuming that the encryption and decryption key are the same. This does not decrease the scope of the definition from the one mentioned before. Recall that when the decryption key K' is different from the encryption key K , K' can be derived from K . Hence the derivation of K' can be included in the decryption function and hence Definition 1.1.1 also includes the case when $K' \neq K$. To simplify notation, when the other parameter is not used in the encryption, we omit the parameter A .

In 1945, Shannon [6] proposed that there are two main properties of a good symmetric key encryption.

- **Confusion:** Each bit of ciphertext should be affected by all the key bits in a complex manner. This is to avoid the leakage of part of the key from the ciphertext. With more bits being involved in the encryption of each bit and with the involvement being more complex, less information about the key will be leaked from the ciphertext and the information will have less impact in finding the key bits. A common method to have confusion property is to use a non-linear function that affects each output bit with many input bits non-linearly. One example is to use an S-box. S-box is a function with m -bit input and n -bit output such that each output bit is affected non-linearly by the input bits. In an encryption function, the combination of S-box and other components can be used to affect each output bits with key bits in a non-linear manner. The non-linearity is commonly introduced only by the S-box.
- **Diffusion:** Each bit of the plaintext should affect the computation of all ciphertext bits. Alternatively, diffusion can be seen as the property that a small change of a plaintext bits affect as many ciphertext bits as possible. This is to ensure that statistical property of plaintexts is dissipated to as many ciphertext bits as possible. For instance, if the plaintext has some specific header, without diffusion property, attacker can easily identify this header and hence part of the plaintext is leaked. In order to introduce diffusion property, a usual method is to use a permutation function to permute the bits before being fed to some

complex function. Using this way, with enough repetition, each plaintext bit can affect all the ciphertext bits and the statistical property of any substring of the plaintext is distributed in the entire ciphertext.

To make sure an encryption scheme has both confusion and diffusion property, different functions can be used in succession. A cipher built by combining multiple functions is called a product cipher. Furthermore, most block ciphers repeated some combination of functions several times to increase its security. In this case, the cipher is called an iterated cipher and the function being repeated is called the round function. The number of times a round function is repeated in an iterated cipher is called round number. To prevent the symmetry among rounds in an iterated cipher, usually different keys are generated from the secret key for different rounds. This process is called the key scheduling function and the key used for each round is called the round key. Increasing of number of rounds usually increases the security of an iterated cipher. However, this is not true when the round function is weak. On the other hand, a large round numbers means that the efficiency of the cipher is low. So a good cipher should have a good balance between security and efficiency.

Now based on the encryption method, there are two main forms of encryption scheme, namely block cipher and stream cipher.

1.1.1.1 Block Cipher

In a block cipher, the message is partitioned into message blocks of fixed length. Each of the blocks is then fed to the same encryption scheme where the key and possibly the previous message block are used as the parameter of the encryption scheme. The method in which a multiple-block message is encrypted is called mode of operation. The encryption function of a single message block by a block cipher can be seen as a pseudorandom injective function from the set of plaintext blocks of fixed length to the set of ciphertext blocks. Two of the most widely implemented block ciphers are DES [112] and AES [48].

Based on how we construct the round function of a block cipher, there are two families that are most-widely used, Substitution Permutation Network (or SPN for short) and Feistel Cipher.

1. **Substitution Permutation Network.** As discussed above, a good encryption scheme should have both confusion and diffusion property. Furthermore, a typical function that can provide confusion is a substitution function and a typical function that can provide diffusion is a permutation function. A Substitution Permutation Network (or SPN for short) applies this directly. A typical SPN cipher is an iterated cipher of r rounds where in each round, substitution and permutation functions are applied to the state value. Typically, a round function of an SPN can be formally defined as follows:

Definition 1.1.2. Let the block size be $n = n_1 \cdot n_2$ for some positive integers n_1 and n_2 . Let $S : \mathbb{F}_2^{n_1} \rightarrow \mathbb{F}_2^{n_1}$ be an invertible S-Box and M be a permutation matrix over $\mathbb{F}_2$ of size $n \times n$. A Substitution Permutation Network has r rounds. Each round function consists of two layers, substitution layer and permutation layer. In the substitution layer, the n -bit state is partitioned into n_2 sub-blocks of size n_1 bits each. The S-Box S is then applied to each sub-block in parallel. In the permutation layer, the permutation matrix is pre-multiplied to the n -bit state to permute the bits of the state.

A good substitution layer should have a highly non-linear S-box that is invertible. It needs to be non-linear so the key and the plaintext can affect the state value and it is not trivial to extract either the key or the previous state value from the current state value. Moreover, it needs to be invertible so a decryption is possible. Some other criteria that can be used to evaluate a cryptographically secure S-box are strict avalanche criterion (in which changing one bit of input should change all the bits of output with probability $\frac{1}{2}$), bit independence (in which the correlation of any two output bits should be as close to zero as

possible) and balance (in which the number of occurrences of zeros and ones in each bits should be equal).

In the permutation layer, to ensure a good confusion and diffusion, the permutation matrix needs to be able to distribute each S-Box output to as many S-boxes in the next round as possible. This way, when one bit of the input is changed, it affects more bits of the later state as soon as possible.

Some widely known SPN are AES [48] and Present [42].

2. **Feistel Cipher.** Feistel cipher was named after Horst Feistel. A precise definition of a Feistel Cipher can be found in the following definition.

Definition 1.1.3. [34] Let P be a $2n$ -bit plaintext, K be the secret key and r be the number of rounds. Partition the plaintext to two n -bit halves, $P = L||R$. We further denote the input of the i -th round function to be $L_i||R_i$ where i is indexed from 0. Hence $L = L_0$ and $R = R_0$. Given a keyed map corresponding to round i , $F_K^{(i)} : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$, round i of an r -round Feistel cipher is defined as follows:

$$\left. \begin{aligned} R_{i+1} &= L_i \\ L_{i+1} &= R_i \oplus F_K^{(i)}(L_i) \end{aligned} \right\} \quad (1.1.1)$$

Then the ciphertext of P is $C = L_r||R_r$. The illustration of round i of an r -round Feistel Cipher is given in Figure 1.1

In most cases, $F_K^{(i)}$ only differs in the key used in the respective round. Given the master secret key, for each round, a key scheduling function is performed to extract a round key from the master secret key. Given this round key, K_i , $F_K^{(i)}$ then performs the same keyed function on the state value with different key parameters.

Figure 1.1: Round i of Feistel cipher

It is easy to see that the decryption process also have r rounds where the i -th round maps the input $(L_{r-i}||R_{r-i})$ to

$$\left. \begin{aligned} L_{r-i-1} &= R_{r-i} \\ R_{r-i-1} &= L_{r-i} \oplus F_K^{(r-1-i)}(L_{r-i}) \end{aligned} \right\} \quad (1.1.2)$$

Many cryptosystems have been constructed based on the Feistel structure, such as DES, Blowfish [124], Kasumi [1] and Camellia [10].

Generalized Feistel Network. Based on the definition of Feistel cipher, more structures have been proposed to generalize the Feistel cipher [111, 125]. For the purpose of this thesis, we introduce 5 different generalization possibilities. We use the term Generalized Feistel Network, or GFN for short, to denote these constructions in general. The main idea of the generalization is based on the partition of the plaintext block to the sub-blocks. In the classical Feistel cipher, we see that the plaintext block is partitioned into two sub-blocks of the same length. For our discussion, the number of sub-blocks the plaintext block is being partitioned to is called the number of branch of the scheme, denoted by b .

A Generalized Feistel Scheme of branch b with n -bit sub-blocks is defined as a (keyed)-permutation $\Pi : (\mathbb{F}_{2^n})^b \rightarrow (\mathbb{F}_{2^n})^b$.

- Type-1 Feistel Schemes. Π is an r -round Type-1 Feistel scheme if Π consists of r repetitions of $\mu_1 : (\mathbb{F}_{2^n})^b \rightarrow (\mathbb{F}_{2^n})^b$ where $\mu_1(x_0, \dots, x_{b-1}) = (x_1 \oplus F_i(x_0), x_2, \dots, x_{b-1}, x_0)$. Assume that $F_i : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_{2^n}$ is a function from n -bit to n -bit which may vary depending on the round where it is being called. Here $i = 1, \dots, r$. Illustration of round i of Type-1 Feistel Scheme can be found in Figure 1.2.

Figure 1.2: Round i of Type-1 Feistel Scheme

- Type-2 Feistel Schemes. For this type, assume that b is even and denote $b = 2\ell$. Then an r -round Type-2 Feistel Scheme Π repeats $\mu_2 : \mathbb{F}_{2^n}^b \rightarrow \mathbb{F}_{2^n}^b$ r times where $\mu_2(x_0, \dots, x_{b-1}) = (x_1 \oplus F_{(i,0)}(x_0), x_2, x_3 \oplus F_{(i,1)}(x_2), \dots, x_{k-1} \oplus F_{(i,\ell-1)}(x_{b-2}), x_0)$. As before, $F_{(i,j)}$ is a function from n -bit strings to n -bit strings with parameter round number i and counter within the round j . Illustration of round i of Type-2 Feistel Scheme can be found in Figure 1.3.
- Type-3 Feistel Schemes. Π is an r -round Type-3 Feistel scheme if Π consists of r iterations of $\mu_3 : \mathbb{F}_{2^n}^b \rightarrow \mathbb{F}_{2^n}^b$. Given $(x_0, \dots, x_{b-1})$, μ_3 maps $(x_0, \dots, x_{b-1})$ to $(x_1 \oplus F_{(i,0)}(x_0), x_2 \oplus F_{(i,1)}(x_1), \dots, x_{b-1} \oplus F_{(i,b-2)}(x_{b-2}), x_0)$. Illustration of round i of Type-3 Feistel Scheme can be found in Figure 1.4.

Figure 1.3: Round i of Type-2 Feistel Scheme

Figure 1.4: Round i of Type-3 Feistel Scheme

- Alternating Feistel Schemes. For this scheme, consider two different round functions $\mu_{A,0}, \mu_{A,1} : \mathbb{F}_{2^n}^b \rightarrow \mathbb{F}_{2^n}^b$ which are used alternatingly for each round.

– $\mu_{A,0}(x_0, \dots, x_{b-1}) = (x_0 \oplus F_i(x_1, \dots, x_{b-1}), x_1, \dots, x_{b-1})$ where $F_i : \mathbb{F}_{2^n}^{b-1} \rightarrow \mathbb{F}_{2^n}$ is called in round $2i - 1$. $\mu_{A,0}$ is called the contracting round.

– $\mu_{A,1}(x_0, \dots, x_{b-1}) = (x_0, x_1 \oplus F_{(i,1)}(x_0), \dots, x_{b-1} \oplus F_{(i,b-1)}(x_0))$. Here $F_{i,j} : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_{2^n}$ is the function called in the j -th component in round $2i$. These rounds are called the expanding rounds. Illustration of round

$2i - 1$ and $2i$ of Alternating Feistel Scheme can be found in Figure 1.5.

Note that round number and index i starts from 1 instead of 0.

Alternatively, $\mu_{A,1}$ can be used in odd rounds and $\mu_{A,0}$ in even rounds but in our discussion a contracting round is always used at round 1.

Figure 1.5: Round $2i - 1$ and Round $2i$ of Alternating Feistel Scheme

- **Unbalanced Feistel Schemes.** This is a special case of the UFN defined in Figure 1 of [74]. Let $b' = 1, \dots, b - 1$ and $F_s : \mathbb{F}_{2^n}^{b-b'} \rightarrow \mathbb{F}_{2^n}^{b'}$ be a map from $(b - b')n$ bit to $b'n$ bit with component functions denoted as $F_{s,0}, \dots, F_{s,b'-1}$ with the round number s as its parameter. Then Π is an r -round UFN(b', b) if it contains r repetitions of $\mu_U : \mathbb{F}_{2^n}^b \rightarrow \mathbb{F}_{2^n}^b$. In round s , given an input $(x_0, \dots, x_{b-1})$, μ_U maps it to $(x_{b'}, \dots, x_{b-1}, x_0 \oplus F_{s,0}(x_{b'}, \dots, x_{b-1}), \dots, x_{b'-1} \oplus F_{s,b'-1}(x_{b'}, \dots, x_{b-1}))$. Illustration of round s of UFN(b', b) is given in Figure 1.6.

3. Comparison between SPN and Feistel Network. In the previous two parts, we discussed two main forms of the round function of a block cipher, Substitution Permutation Network and Feistel Network. In this part, we will discuss the difference of the two and the advantage and disadvantage of each

Figure 1.6: Round s of Unbalanced Feistel Scheme

form. As mentioned in the introduction of SPN, an SPN is built by simply interleaving substitution (for confusion) and permutation (for diffusion) layer. Due to its simple philosophy, it is actually easier to conduct the analysis of the security of SPN. An easy analysis provides us with quite accurate lower and upper bound of its security and hence we can provide a better choice of number of rounds that has a good balance between efficiency and security.

On the other hand, due to its design, for the cipher to be invertible, we need both substitution and permutation layer to be invertible. Because of this there are more restriction in the choice of substitution and permutation layer. Furthermore, from the implementation point of view, both encryption and decryption of the cipher needs to be implemented to make sure both encryption and decryption of the cipher can be applied. This is not true in the Feistel structure. As discussed before, the decryption of a Feistel structure in fact has almost identical structure with its encryption. So the implementation of the encryption can directly be used for the decryption, with some change of the key schedule, namely flipping the order of the round keys. Since in the decryption process we do not really need to use the inverse of the round function F , F does not have to be invertible.

In summary, although SPN is easier to analyze, it may need more area in the hardware implementation since we need to implement both encryption and

decryption process of the cipher. This also means that all layers of SPN needs to be invertible. On the other hand, in a Feistel structure, the implementation of the encryption process can be used for the decryption process with the order of the key schedule being inverted. This gives us two advantages, the hardware implementation can be more efficient and the round function F does not need to be invertible.

1.1.1.2 Stream Cipher

In this section, we mainly discuss the synchronous stream cipher. Different from a block cipher, in a synchronous stream cipher, the secret key is used to generate a pseudorandom key bit stream of the same length of the plaintext. The encryption is done by XORing the key stream with the plaintext. Since the keystream is generated independently from the plaintext, once we know a plaintext-ciphertext pair encrypted using one fixed secret key K , the value of the key stream bits generated by K can be recovered. Because of this, even if we do not have the secret key, with the knowledge of a plaintext-ciphertext pair, we can easily encrypt any message with the same secret key and decrypt any ciphertext encrypted using the same secret key. In order to solve this problem, a synchronous stream cipher uses an Initialization Vector (IV) to make sure every key stream generation produces different streams even when the secret key is the same. Some popular stream ciphers includes RC4 and ChaCha [19].

In comparison, in an asynchronous stream cipher, the previous plaintext bits affects the generation of the current key-stream bits. Because of this, a nonce-reuse will not leak the whole key-stream as long as the plaintext that is being encrypted is not exactly the same.

First we note that it is possible to construct a stream cipher using a block cipher. One example is to use a mode of operation called the output feedback mode (OFB). As in the stream cipher, it requires the use of a IV/nonce. The idea is that we use the nonce as the plaintext for the block cipher. Then the ciphertext is used

as the keystream for the first message block and for all the other message blocks, the keystream of the previous message block is used as the plaintext of the current computation of the block cipher to generate the keystream of the current message block.

Next we discuss one advantage of block cipher over synchronous stream cipher. Due to the involvement of the message in the encryption function, each bit of the input affects each other in the calculation of the output. This is not true in the stream cipher since the message is only XORed to the key stream generated by the encryption function. One way to solve this problem is through the use of nonce. However, the use of nonce in a synchronous stream cipher needs some scheme to generate the nonce in the appropriate way to avoid the repetition of nonce. Although in a block cipher, the use of nonce can also be necessary (for example, in a CBC mode of operation), nonce reuse in a block cipher, as long as it is not in the OFB mode, that requires a nonce will not make the block cipher unusable. Although it leaks some information about the encryption function, it does not reveal all the information. On the other hand, in a synchronous stream cipher, if a key-nonce pair is used, the keystream is revealed and anyone without the key can encrypt any other message using the same key-nonce pair. Similarly, when a plaintext is encrypted using the same key-nonce pair, anyone without the key can easily decrypt it.

On the other hand, stream cipher also has some advantages over block cipher. Firstly, it is designed to be able to encrypt plaintexts of arbitrary length with less increase of complexity. For example, assuming the block size of a block cipher is n -bits. When the plaintext is increased from n to $n + 1$ bits, this block cipher will require two different calls of the block cipher. On the other hand, if we use a stream cipher, it just means that we need to generate one more bit of keystream. Due to this, it is more suitable for online encryption when the message packet is encrypted during transmission.

Secondly, in a synchronous stream cipher, the fact that the plaintext is not required

to generate the key-stream means that we can precompute the key stream before we have the plaintext. Having this key stream computed offline means that when we have the plaintext, the encryption is simply just XOR of two strings which can be done very efficiently.

1.1.2 Hash Function

A hash function is a cryptographic primitive that is designed to be used in constructing digital signature, MAC and key derivation for other cryptosystems. A hash function is typically an unkeyed function H with arbitrary input length and a fixed length output. An ideal hash function satisfies three security requirements:

- Pre-image resistance: Given the message digest $h = H(m)$ of a message m , it is computationally infeasible to find m such that $h = H(m)$.
- Second Pre-image resistance: Given m and its hash value $h = H(m)$, it is computationally infeasible to find another message $m' \neq m$ such that $H(m') = h$.
- Collision resistance: It is computationally infeasible to find two different messages m and m' such that $H(m) = H(m')$.

A compression function of fixed input and output length is usually used to construct a hash function that can accept any input length. A hash function construction based on this idea is called an iterated hash function. We will shortly describe the main idea of iterated hash function.

1.1.2.1 Iterated Hash Function

In order to describe a hash function construction from a compression function, we first give some notations that will only be used in this section.

Let $\mathbf{c} : \mathbb{F}_2^{m+t} \rightarrow \mathbb{F}_2^m$ be a compression function and $\mathbf{h} : \mathbb{F}_2^\star \rightarrow \mathbb{F}_2^\ell$ be a hash function where $\mathbb{F}_2^\star = \bigcup_{i \geq 0} \mathbb{F}_2^i$. Note that the message digest length may not be the same as the output length of $\mathbf{c}$.

Given the input x of $\mathbf{h}$, we first pad x by a function $\mathbf{p} : \mathbb{F}_2^\star \rightarrow \bigcup_{i \geq 1} \mathbb{F}_2^{t' \cdot i}$ to a string of length rt' for some positive integers r and $t' \leq t$. After being padded to $x' = \mathbf{p}(x)$, partition it to substrings of length t' each, $x' = x_1 || \dots || x_r$. We also require some publicly known IV h_0 of length $m' \leq m$. Then the message digest is recursively computed by the formula $h_i = \mathbf{c}(\mathbf{f}(x_i, h_{i-1}))$ for $i = 1, \dots, r$ where $\mathbf{f}$ is some process to transform x_i and h_{i-1} to an $m + t$ -bit string for the input of $\mathbf{c}$. We then use a publicly known function $\mathbf{g} : \mathbb{F}_2^m \rightarrow \mathbb{F}_2^\ell$ such that the message digest $h = \mathbf{g}(h_r)$.

In order to make the hash function $\mathbf{h}$ computable by anyone and also secure, there are some requirements for $\mathbf{c}$, $\mathbf{f}$ and $\mathbf{p}$. First of all, $\mathbf{c}$ needs to also have a collision resistance property. Otherwise, it will not be possible for $\mathbf{h}$ to be collision resistant. The padding and combining function $\mathbf{p}$, $\mathbf{f}$ need to be carefully designed such that given two different input x, y , $\mathbf{p}(x) \neq \mathbf{p}(y)$. In particular, $\mathbf{p}$ needs to be injective. Due to these requirements, it is interesting to study how the padding and the iterative relation can be designed.

One specific construction of an iterated hash function that is widely used is the Merkle-Damgård Construction [46]. The advantage of using the Merkle-Damgård Construction is that formal security proof of any hash function based on the Merkle-Damgård Construction can be provided.

It can be proved that a hash function $\mathbf{h}$ based on Merkle-Damgård construction and compression function $\mathbf{c}$ is collision resistant if the compression function $\mathbf{c}$ also has the collision resistance property [128]. SHA-1 [8] and SHA-2 [3] are two widely used hash functions that were designed based on the Merkle-Damgård construction.

1.1.3 Authentication Encryption Scheme

In some of the real life applications, a message transmission requires confidentiality, integrity and authenticity at the same time. Authenticated Encryption (AE) scheme was first proposed by Bellare and Namprempre [16] in 2000 to fulfill this need. In the application, part of the message such as the routing information cannot be encrypted. However, this part of the message needs to also be authenticated. For this, Authenticated Encryption with Associated Data (AEAD) was proposed to extend the notion of AE by Rogaway [122].

Mathematically, AEAD can be defined as follows.

Definition 1.1.4. Let $\mathcal{P}, \mathcal{K}, \mathcal{N}, \mathcal{A}, \mathcal{C}$ and $\mathcal{T}$ be space of plaintext, secret keys, nonce/IV, associated data, ciphertext and authenticated tags respectively.

An Authenticated Encryption with Associated Data with secret key $K \in \mathcal{K}$ is a pair of functions, $E_K : \mathcal{N} \times \mathcal{A} \times \mathcal{P} \rightarrow \mathcal{C} \times \mathcal{T}$ and $D_K : \mathcal{N} \times \mathcal{A} \times \mathcal{C} \times \mathcal{T} \rightarrow \{\perp\} \cup \mathcal{M}$ such that:

1. If $E_K(N, A, P) \neq (C, T) \forall (N, A, P) \in \mathcal{N} \times \mathcal{A} \times \mathcal{P}, D_K(C, T) = \perp$.
2. If $E_K(N, A, P) = (C, T)$, we have $D_K(N, A, C, T) = P$.

The first requirement given in Definition 1.1.4 requires that if T is not a valid tag for ciphertext C , nonce N and associated data A , then the function must return $\perp$ which means it will not reveal the value of the calculated P . It can be seen that an AEAD provides confidentiality of the message and the tag provides authenticity for message and associated data.

Three main designs were considered in [16]:

1. Encrypt-then-MAC: In this design, the plaintext is encrypted, then a MAC is applied to the ciphertext to provides authenticity of the ciphertext. It was applied in ISO/IEC 19772:2009 [4].

2. MAC-then-Encrypt: In this design, contrary to the previous design, the plaintext is being authenticated first then encryption is done to both the plaintext and the tag. SSL/TSL uses this for its communication protocol.
3. Encrypt and MAC: Plaintext is used for the input of both the encryption scheme and the MAC.

Utilising existing encryption schemes and MAC to construct AE must be done carefully. Otherwise, the AE may be broken even if both the encryption schemes and MAC are secure. Two examples of this are BEAST [60] and POODLE [100] attack on SSL-3. In both attacks, the plaintext can be recovered even if the secret key is unknown. The attacks do not even consider the detail of the encryption scheme and MAC. So even when both schemes used are ideal, the protocol will still be insecure. Aside from a possible security issue, the efficiency of AE constructed by combining two existing constructions is also an issue. This is because for one AE, we would need to implement two primitives, one encryption scheme and one MAC.

One better way to construct an AEAD is to have a carefully-designed block cipher mode of operation to combine the encryption and authentication from the existing cryptographic primitives. Some of the mode of operations that enables both encryption and authentication at the same time are ELM [52], CCM [61], GCM [62], IAPM [82], OCB [120] and JAMBU [136].

Another way is to construct a dedicated authenticated encryption with associated data based on other existing cryptosystem which is utilised to provide both confidentiality and authentication. Some of the examples of a dedicated authenticated encryption with associated data based on other existing cryptosystem are AEGIS [73] and Tiaoxin [107].

An AEAD can be designed to be a stream cipher-based authenticated encryption scheme. Some examples of dedicated AEAD that is based on stream cipher are Hummingbird-2 [63], Helix [66], ASC-1 [80] and Phelix [135]. Alternatively, AEAD can also be hash function-based. Some of them includes OMD [45] and COFFE [67].

COFFE will be analyzed further in Chapter 5

1.2 Cryptanalysis of Symmetric Key Cryptosystem

Kerckhoffs' Principle states that the security of a cryptosystem should be measured based on the assumption that the attacker, Eve, knows all the information about the cryptosystem except for the secret key. The security of a cryptosystem can be measured differently based on the cryptosystem. One of the mostly-used methods (besides security proof) to measure the security of a cryptosystem is through measuring the complexity of different attacks performed on it. In general, any cryptosystem is expected to behave like a random permutation to prevent any information leakage about the plaintext and the secret key. One of the attacks that is applicable to any cryptosystem is to distinguish the output of a cryptosystem from a random permutation. This attack is called the distinguishing attack. Although this attack by itself seldom provides the attacker with much information about the secret key to recover it, a distinguishing attack may be extended to a more advanced attack.

In an encryption scheme, there are mainly four other types of attacks that we can consider:

- **State Recovery Attack.** This attack aims to deduce some secret information of the state value during encryption (or decryption) that should not be accessible before.
- **Key Recovery Attack.** One of the most powerful attacks that can be launched against an encryption scheme is the attack that can reveal the value (or partial value) of the secret key. Once the secret key is revealed, similar to the effect of a global forgery attack, the attacker can encrypt and decrypt all the plaintext and ciphertext respectively.

The main focus of attack for hash function is to find a collision in the function. With the birthday attack [69], the complexity finding a collision with probability

above 0.5 is $2^{\frac{h}{2}}$ where h is the length of the message digest. So a secure hash function should have that as the complexity to find a collision.

Attacks for MAC have similar objective as the other attacks. Aside from the distinguishing attack and the key recovery attack, another objective of attacks on MAC is to be able to forge a message tag without knowing the secret key. This can be classified into two groups based on the number of messages whose tag we can forge: local forgery attack, if the set of messages we can forge is just a subset of $\mathcal{P}$ and global forgery attack, if we can forge the tag of all possible messages.

In this thesis, we focus on the attacks of symmetric key encryption and AEAD.

There are mainly four basic attack models in analysing the security of a cryptosystem:

1. Ciphertext only attack. This is the weakest model of an attack. In this model, the attacker, Eve, is assumed to have access to ciphertexts encrypted by the cryptosystem being analyzed.
2. Known plaintext attack. Different from the ciphertext only attack, in this model, Eve is assumed to have access to some plaintext-ciphertext pairs (P_i, C_i) for $i = 1, \dots, q$ for some positive integer q that she can use to launch the attacks mentioned above. This is a more powerful attack model than the ciphertext only attack since now Eve knows the plaintext that is encrypted to the ciphertext she has.
3. Chosen plaintext attack. In this model, instead of having q random plaintext-ciphertext pairs to use, Eve is assumed to have access to some encryption oracle that she can use to encrypt q plaintexts of her choice. Again, this attack model is more powerful than the known plaintext attack since now the plaintexts can be chosen to reveal as much information as possible.
4. Chosen ciphertext attack. In addition to the Chosen plaintext attack where Eve has an access to the encryption oracle, in this model, she also has an access to the

decryption oracle. So in this model, Eve can have q_1 plaintexts to be encrypted by the encryption oracle and q_2 ciphertexts to be decrypted by the decryption oracle. Obviously, this shows that this model has a stronger assumption than the chosen plaintext attack.

In some of the analysis, to recover the value of secret key, we can assume that the attacker can obtain some information about the encryption (or decryption) with some other key that has some relation with the original secret key. In this setting, it is called related-key attack. Related-key attack has been considered in cryptanalyzing KASUMI [106] and AES-256 [35].

In the following section, we review some of the existing attacks on symmetric key cryptosystems that will be useful in the following part of this thesis.

1.2.1 Linear Cryptanalysis

Linear Cryptanalysis was first introduced by Matsui and Yamagishi [97] to attack FEAL. The main idea of the attack is to find a linear approximation of the cipher. This is done by finding a linear relation involving plaintext bits, ciphertext bits and key bits with a significant bias. If the bias is high enough, this can be used as a distinguisher between the encryption scheme with a random permutation.

Formally, suppose that we want to analyze the function $f : \mathbb{F}_2^m \rightarrow \mathbb{F}_2^n$. Then a linear characteristic for f is a pair $(\Gamma_I, \Gamma_O) \in \mathbb{F}_2^m \times \mathbb{F}_2^n$. The bias of a linear characteristic (Γ_I, Γ_O) is formulated as the distance between $\frac{1}{2}$ and the $p.p$ is defined to be the probability that for an input $x \in \mathbb{F}_2^m$ such that $\langle x, \Gamma_I \rangle = 0$ to be mapped to $f(x)$ such that $\langle f(x), \Gamma_O \rangle = 0$. Here $\langle ., . \rangle$ denotes the standard inner product of vectors over $\mathbb{F}_2$. So in summary, the bias ϵ of a non-zero linear characteristic (Γ_I, Γ_O) is

$$\begin{aligned}
\epsilon &= \left| \frac{1}{2} - \Pr_{x \in \mathbb{F}_2^m, \langle x, \Gamma_I \rangle = 0}(\langle f(x), \Gamma_O \rangle = 0) \right| \\
&= \left| \frac{1}{2} - \frac{|\{x \in \mathbb{F}_2^m : \langle x, \Gamma_I \rangle = \langle f(x), \Gamma_O \rangle = 0\}|}{|\{x \in \mathbb{F}_2^m : \langle x, \Gamma_I \rangle = 0\}|} \right| \\
&= \left| \frac{1}{2} - \frac{1}{2^{m-1}} \cdot |\{x \in \mathbb{F}_2^m : \langle x, \Gamma_I \rangle = \langle f(x), \Gamma_O \rangle = 0\}| \right|.
\end{aligned}$$

Here the last equation comes from the fact that the solution space of $\langle x, \Gamma_I \rangle = 0$ is a dimension $m - 1$ subspace of $\mathbb{F}_2^m$ for any non-zero Γ_I .

Next we discuss how we can use this distinguisher to recover more information about the secret key. We assume that the attack model is at least in the known plaintext attack model.

In general, when launching either linear cryptanalysis, differential cryptanalysis or their variants, the following assumption is made.

Assumption 1.2.1. [90] Any inputs and round keys used in any rounds are assumed to be independent and uniformly distributed. Furthermore, for any given round keys, the differential or linear probability is assumed to be approximately equal

Typically, the extension of this distinguisher to a key recovery attack can be done by finding a linear relation between bits from the input and output of the encryption scheme. As mentioned before, the bias of the linear relation chosen should be high enough. Given this linear relation, we can propagate linear relation of the input and the output for some extra rounds to a linear relation of the plaintext bits, key bits and ciphertext bits. This can be used to recover the key bits. Given this linear relation, we collect T plaintext-ciphertext pairs. For each of the key guesses, we count the number of pairs that satisfy the linear relation given above.

Assuming that when the key guess is wrong the reduced version becomes a random permutation, the correct key guess should have the largest number of pairs that satisfies the linear relation mentioned above. However, this may not always be true in the application when the number of pairs T is small. Given that the bias is ϵ ,

a successful attack on the cipher using this linear relation needs to have $T = c\epsilon^{-2}$ plaintext-ciphertext pairs for some small constant c [128]. This is to make sure that the probability of a wrong conclusion in the correct key guess is low enough.

1.2.2 Differential Cryptanalysis

Differential Cryptanalysis was first introduced by Biham and Shamir [29, 30]. The idea is to investigate the impact of difference introduced in the input to the difference of the output. In a random permutation, for any non-zero input differences, all the non-zero output differences must have the same probability. So we would like to find a differential path, with a specific input difference such that the probability of the specified output difference to happen is high enough. Mathematically we can define the differential probability as follows

Definition 1.2.2. Consider a function $f : \mathbb{F}_2^m \rightarrow \mathbb{F}_2^n$. Let $\delta_I \in \mathbb{F}_2^m, \delta_O \in \mathbb{F}_2^n$ be the input and output differences respectively. Then the differential probability of $\delta_I \rightarrow \delta_O$ is the probability that two input strings of difference δ_I is mapped to a pair of outputs with difference δ_O . Assuming that the input is uniformly distributed, the differential probability is

$$\begin{aligned} p_{\delta_I \rightarrow \delta_O} &= Pr_{(x_1, x_2) \in (\mathbb{F}_2^n)^2, x_1 \oplus x_2 = \delta_I} (f(x_1) \oplus f(x_2) = \delta_O) \\ &= \frac{1}{2^m} \cdot |\{x \in \mathbb{F}_2^m : f(x \oplus \delta_I) = f(x) \oplus \delta_O\}| \end{aligned}$$

Suppose that f is a bijection. In particular, we can assume that $m = n$. Since f is a bijection, we can find $g : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ such that for any $x \in \mathbb{F}_2^n, f \circ g(x) = g \circ f(x) = x$. Now consider the differential $\delta_O \rightarrow \delta_I$ in g . Let

$$S_1 = \{x \in \mathbb{F}_2^n : f(x \oplus \delta_I) = f(x) \oplus \delta_O\}$$

and

$$S_2 = \{y \in \mathbb{F}_2^n : g(y \oplus \delta_O) = g(y) \oplus \delta_I\}.$$

It is easy to see that $x \in S_1 \leftrightarrow f(x) \in S_2$. So $|S_1| = |S_2|$ and hence the differential probability of $\delta_I \rightarrow \delta_O$ for f is the same as the differential probability of $\delta_O \rightarrow \delta_I$ for its inverse g . Similar to the discussion in the linear cryptanalysis, in the differential cryptanalysis, when the key guess is wrong, the encryption scheme should behave like a random permutation. So, the correct key guess should yield the most number of pairs of inputs such that the difference of their outputs follows the specified differences when the data size is large enough. If the differential probability is ϵ , the number of input pairs should be at least $c\epsilon^{-1}$ for some small constant c .

Differential cryptanalysis is widely used to analyse the security of a symmetric key cryptography. Here the difference is in the input and the output instead of the plaintext and ciphertext. This is because the difference in input can be introduced not only in the plaintext. The input difference, alternatively, can also be introduced in IV. In AEAD, the input difference can be introduced in associated data as well.

There have been several variants of differential cryptanalysis that enable for a more advanced cryptanalysis. In the remaining part of this subsection, we discuss three specific variants of differential cryptanalysis that we will be using in this thesis, namely truncated differential analysis, impossible differential analysis and boomerang attack.

1.2.2.1 Truncated Differential Cryptanalysis

Truncated differential cryptanalysis is a variant of differential cryptanalysis that was introduced by Knudsen in 1994 [84]. In the original differential cryptanalysis, the input and output differences are fixed values. The main idea of truncated differential cryptanalysis is to loosen this requirement. Instead of requiring to specify the differences for all n -bits of input and n -bits of output, we only require a certain portion of bits where the differences must happen.

More specifically, we first partition the plaintext and ciphertext to sub-strings. Suppose that an input string is partitioned to n'_1 sub-strings of length ℓ_1 each while

an output string is partitioned to n'_2 sub-strings of length ℓ_2 each. This can be further generalised by allowing the sub-strings to have different lengths. Having this partition, a truncated differential specifies the sub-strings of plaintext and ciphertext which are non-zero. A truncated differential of a cryptosystem $f : \mathbb{F}_{2^{\ell_1}}^{n'_1} \rightarrow \mathbb{F}_{2^{\ell_2}}^{n'_2}$ can be expressed as a pair of sets (S_I, S_O) where $S_I, S_O \subseteq \{1, \dots, n'_1\}$. An input pair $((I_{1,1}, \dots, I_{1,n'_1}), (I_{2,1}, \dots, I_{2,n'_2}))$ with valid difference belonging to S_I satisfies the following difference; For any $j \in \{1, \dots, n'_1\}$,

$$\begin{aligned} I_{1,j} \oplus I_{2,j} &= 0, \text{ if } j \notin S_I \\ I_{1,j} \oplus I_{2,j} &\in \mathbb{F}_{2^{\ell_2}} \setminus \{0\}, \text{ if } j \in S_I. \end{aligned}$$

A valid output pair can also be defined in the same way. Thus, we have a set of valid input differences $\Delta_I \subseteq \mathbb{F}_2^m$ and a set of valid output differences $\Delta_O \subseteq \mathbb{F}_2^n$. Given these sets, we can define the truncated differential probability as follows.

Definition 1.2.3. Let $f : \mathbb{F}_2^m \rightarrow \mathbb{F}_2^n$ be a function. Let $\Delta_I \subseteq \mathbb{F}_2^m$ and $\Delta_O \subseteq \mathbb{F}_2^n$ be the input and output differential. Then the truncated differential probability of $\Delta_I \rightarrow \Delta_O$ is the probability that a pair of input with difference belonging to Δ_I is mapped by f to output pair of difference belonging to Δ_O . Assuming that the input is uniformly distributed, the differential probability is

$$\begin{aligned} p_{\Delta_I \rightarrow \Delta_O} &= Pr_{(x_1, x_2) \in (\mathbb{F}_2^n)^2, x_1 \oplus x_2 \in \Delta_I} (f(x_1) \oplus f(x_2) \in \Delta_O) \\ &= \frac{1}{2^m \cdot |\Delta_I|} \cdot |\{(x, \delta_I) \in \mathbb{F}_2^m \times \Delta_I : f(x \oplus \delta_I) \oplus f(x) \in \Delta_O\}| \end{aligned}$$

Suppose f is a bijection and g its inverse. Let

$$S_1 = \{(x, \delta_I) \in \mathbb{F}_2^m \times \Delta_I : \exists \delta_O \in \Delta_O, f(x \oplus \delta_I) = f(x) \oplus \delta_O\}$$

and

$$S_2 = \{(y, \delta_O) \in \mathbb{F}_2^n \times \Delta_O : \exists \delta_I \in \Delta_I, g(y \oplus \delta_O) = g(y) \oplus \delta_I\}.$$

In standard differential cryptanalysis, $(x, \delta_I) \in S_1$ if and only if $(f(x), f(x) \oplus f(x \oplus \delta_I)) \in S_2$. So $|S_1| = |S_2|$. However, this does not mean that the differential probability

of $\Delta_I \rightarrow \Delta_O$ for f is necessarily the same as the differential probability of $\Delta_O \rightarrow \Delta_I$ for g since the size of Δ_I and Δ_O may not be the same. A simple example is if we let $\Delta_O = \mathbb{F}_2^n$ and $\Delta_I = \{\delta_I\}$ for some $\delta_I \in \mathbb{F}_2^n$. Let p_1 be the differential probability of $\Delta_I \rightarrow \Delta_O$ for f and p_2 be the differential probability of $\Delta_O \rightarrow \Delta_I$ for g . Now given any $x \in \mathbb{F}_2^n$, $f(x) \oplus f(x \oplus \delta_I) \in \mathbb{F}_2^n = \Delta_O$. So $|S_1| = |S_2| = 2^n$. Hence $p_1 = 1$. However, $p_2 = \frac{1}{2^n \times 2^n} \cdot 2^n = \frac{1}{2^n} < 1 = p_1$.

Truncated differential cryptanalysis has been applied to cryptanalyze Salsa20 [47], Skipjack [85], SAFER [86, 140] and Camellia [91].

1.2.2.2 Impossible Differential Cryptanalysis

Impossible differential cryptanalysis was introduced by Lars Knudsen to cryptanalyze DEAL [83]. Instead of finding an output difference that has some probability to occur given an input difference, we consider the input-output difference pair that is impossible to happen. Using this, if by using some key guess, this input-output difference occur, then we can be certain that the key guess is wrong.

Impossible differential cryptanalysis has been used to analyse many designs including AES [94] and Skipjack [25]. It is also used to analyze the security of a general Feistel Cipher with SP and SPS round function [133]. As we will see later, impossible differential cryptanalysis can be combined with the truncated differential attack.

1.2.2.3 Boomerang Attack

Boomerang Attack was first introduced by David Wagner to break COCONUT98 [132]. When the round number g rows, a standard differential cryptanalysis may not be able to provide one differential path for the whole round. The idea behind a boomerang attack is to combine two different differential paths, each of which is for a part of the function to analyse. Suppose that the function that we want to analyse is the bijection $E_K : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$. Let $E_0 : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ and $E_1 : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ be two bijections such that $E_K = E_1 \circ E_0$. Suppose that we have a differential path $\delta_I \rightarrow \delta_O$ for E_0 with

differential probability $p_0 \in (0, 1]$ and $\gamma_I \rightarrow \gamma_O$ for E_1 with differential probability $p_1 \in (0, 1]$.

Consider a pair of input P_1 and P_2 of E_0 such that $P_1 \oplus P_2 = \delta_I$. Let $Q_i = E_0(P_i)$ and $R_i = E_K(P_i)$ for $i = 1, 2$. Then $Q_1 \oplus Q_2 = \delta_O$ with probability p_0 . Now consider R_3, R_4 such that $R_1 \oplus R_3 = R_2 \oplus R_4 = \gamma_O$. Let $Q_i = E_1^{-1}R_i$ and $P_i = E_K^{-1}R_i$ for $i = 3, 4$. With probability p_1^2 , $Q_1 \oplus Q_3 = Q_2 \oplus Q_4 = \gamma_I$. Assuming $Q_1 \oplus Q_2 = \delta_O$, we have $Q_3 \oplus Q_4 = \delta_O$. With probability p_1 , $P_3 \oplus P_4 = \delta_I$. With two encryption queries and two decryption queries, when the function f is not a random function, with probability of at least $p_0^2 \cdot p_1^2$, the second input pairs has difference δ_I .

Here the probability of the differences to hold is at least $p_0^2 \cdot p_1^2$ since for any δ_O and γ_I can be changed for any possible values and it will increase the probability.

This attack can be generalised by using truncated differentials in both halves of the function. Suppose that $\Delta_I \rightarrow \Delta_O$ is a truncated differential for E_0 of probability $p_0^{(F)}$ in the forward direction and $p_0^{(B)}$ in the backward direction. Suppose further that $\nabla_I \rightarrow \nabla_O$ is a truncated differential for E_1 of probability $p_1^{(F)}$ and $p_1^{(B)}$ in the forward and backward direction respectively. The probability can be computed by the following lemma.

Lemma 1.2.4. [132] *Let E be a bijection that can be decomposed to $E = E_1 \circ E_0$. Let $\Delta_I \rightarrow \Delta_O$ be a truncated differential characteristic for E_0 with forward and backward probabilities $p_0^{(F)}$ and $p_0^{(B)}$ respectively. Let $\nabla_I \rightarrow \nabla_O$ be a truncated differential characteristic for E_1 with forward and backward probabilities $p_1^{(F)}$ and $p_1^{(B)}$ respectively.*

Then a truncated boomerang attack using these two characteristics has success probability p where

$$p \approx p_0^{(F)} \cdot (p_1^{(B)})^2 \cdot p_0^{(B)} \cdot \Pr(\gamma_{I,1} \oplus \gamma_{I,2} \oplus \delta_O \in \Delta_O | \delta_O \in \Delta_O, \gamma_{I,1}, \gamma_{I,2} \in \nabla_I).$$

Boomerang attack has been used to analyze the security of COCONUT98, Khufu-16, FEAL-6, CAST-256 [132], AES [33] and KASUMI [28].

1.2.3 Differential-Linear Cryptanalysis

Differential-linear cryptanalysis was introduced by Langford and Hellman in 1994 [90]. It was then improved further by Langford [89], Biham et al [26] and Lu [93]. Similar to the boomerang attack, the idea is to combine two different characteristics of part of the function to analyse the security of an encryption scheme. Instead of having two differential characteristics to be combined, we consider a (truncated) differential characteristic which is followed by a linear characteristic.

In general, we decompose an encryption scheme $E_K : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ to two functions $E_0 : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ and $E_1 : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ such that $E_K = E_1 \circ E_0$. We then find a (truncated) differential characteristic for $E_0, \Delta_I \rightarrow \Delta_O$ with probability p_0 and a linear characteristic $\Gamma_I \rightarrow \Gamma_O$ for E_1 with bias ϵ . We further require that for any $\delta_O \in \Delta_O, \langle \Delta_O, \Gamma_I \rangle = 0$ where $\langle \cdot, \cdot \rangle$ represents the standard inner product of two vectors. In other words, for any pairs Q_0, Q_1 such that $Q_0 \oplus Q_1 \in \Delta_O, \langle Q_0, \Gamma_I \rangle = \langle Q_1, \Gamma_I \rangle$. We are interested to find the probability that $\langle E_K(P_0), \Gamma_O \rangle = \langle E_K(P_1), \Gamma_O \rangle$ for any input pairs (P_0, P_1) such that $P_0 \oplus P_1 \in \Delta_O$.

Along with Assumption 1.2.1, the following assumption is made:

Assumption 1.2.5. [93] Given any pairs of input (P_0, P_1) such that $P_0 \oplus P_1 \in \Delta_I, E_0(P_0)$ and $E_0(P_1)$ behaves as independent inputs for the linear characteristic for E_1 , the second half of E_K .

Initially, in [90], with the assumption that $p_0 = 1$, it can be proven that

$$Pr_{P_0, P_1 \in \mathbb{F}_2^n : P_0 \oplus P_1 \in \Delta_I} (\langle E_K(P_0), \Gamma_O \rangle = \langle E_K(P_1), \Gamma_O \rangle) = \frac{1}{2} + 2\epsilon^2.$$

Hence, when $2\epsilon^2$ is large enough, it can be used to distinguish E_K from a random permutation.

There are some further improvements in [89] and [26]. The main objective of the improvements is to allow p_0 to be less than 1. Firstly, [26] works under the following assumption to get the following result:

Theorem 1.2.6. [93] *First, we consider the following assumption*

Assumption 1.2.7. [93] When $E_0(P_0) \oplus E_0(P_1) \notin \Delta_O$, $\langle E_K(P_0), \Gamma_O \rangle$ and $\langle E_K(P_1), \Gamma_O \rangle$ are uniformly and independently distributed on $\{0, 1\}$.

Under this assumption,

$$Pr_{P_0, P_1 \in \mathbb{F}_2^n : P_0 \oplus P_1 \in \Delta_I} (\langle E_K(P_0), \Gamma_O \rangle = \langle E_K(P_1), \Gamma_O \rangle) = \frac{1}{2} + 2p_0\epsilon^2.$$

Furthermore, the resulting distinguishing attack has complexity $\mathcal{O}(p_0^{-2}\epsilon^{-4})$.

Following this result, it is then improved in [93] by omitting the use of Assumption 1.2.7. The resulting theorem is as follows.

Theorem 1.2.8. [93] *Let $E = E_1 \circ E_0$. Let $\delta_I \in \mathbb{F}_2^n \setminus \{0\}$ be a non-zero input difference and $\Gamma_I \rightarrow \Gamma_O$ is a linear approximation for E_1 with bias ϵ . Let the sum of the probabilities for the differentials for the differentials*

$$\{\delta_I \rightarrow \delta_O : p_{\delta_I \rightarrow \delta_O} = 0, \langle \Gamma_I, \delta_O \rangle = 0\}$$

be

$$\hat{p} = \sum_{\delta_O \in \mathbb{F}_2^n, \langle \Gamma_I, \delta_O \rangle = 0} p_{\delta_I \rightarrow \delta_O}.$$

Then under Assumptions 1.2.1 and 1.2.5, the probability of the differential-linear distinguisher $\delta_I \rightarrow \Gamma_O$ is

$$Pr_{P \in \mathbb{F}_2^n} (\langle E(P), \Gamma_O \rangle = \langle E(P \oplus \delta_I), \Gamma_O \rangle) = \frac{1}{2} + 2(2\hat{p} - 1)\epsilon^2.$$

This attack has been used to analyse the security of several cryptosystems including DES [93], Camellia [145] and Phelix [142].

1.2.4 Security Analysis Related to Padding

Aside from the cryptanalysis described above, it is also possible to exploit the padding protocol of the input before being encrypted to the ciphertext. Some of the known attacks that consider this includes the BEAST attack [60] and POODLE attack [100].

In both attacks, they exploit the padding system and the procedure of encryption and authentication to recover the secret message even if both the encryption scheme and MAC are ideal functions. Some other examples of attack that arises from the problem in the padding procedure can be found in the attack against CMCC mode [14], SCREAM [127] and PiCipher [92].

1.3 Thesis Outline

Chapter 1 contains the review of some concepts that will be useful in the remaining part of the thesis. This includes a brief introduction on some cryptographic primitives. It is then followed by a brief introduction on some cryptanalysis on symmetric key cryptosystem that is focused mainly on differential cryptanalysis.

Chapter 2 contains our analysis on Simpira v2. In this chapter we will be utilising truncated differential Cryptanalysis, impossible differential cryptanalysis and boomerang attack to analyze a version of Simpira v2 with 3 branches. Using these analysis, we managed to find distinguishers and key recovery attacks for 8-,9- and 10-rounds of Simpira v2.

Chapter 3 contains our analysis on Generalized Feistel Scheme. In this chapter we consider a truncated differential for five different types of Generalized Feistel Scheme with any round function. We provide lower bound for the maximum number of rounds that is distinguishable from a random permutation for various designs of Generalized Feistel Scheme.

Chapter 4 contains a differential linear cryptanalysis on a CAESAR candidate ICEPOLE. In this analysis, we are considering the variant where both public initialization vector (or nonce) and secret message number can be reused to encrypt different plaintexts. Under this restriction, we find a distinguisher for the cryptosystem which can be extended to a state recovery attack.

Chapter 5 contains our analysis on the hash-based authenticated encryption scheme

COFFE where the underlying hash function is assumed to be ideal. Note that the analysis is not based on the standard concept of differential cryptanalysis where we consider the difference on two plaintexts and study its effect on the difference on the outputs of the two plaintexts. Instead, we consider a pair of distinct inputs which can lead to the same state value before encryption. Here input includes initialization vector, associated data, block length, last block length and plaintext. In the second part of the chapter, we also consider to analyse the encryption scheme under related key setting in which the two keys have different length and the longer key is obtained by extending the shorter key by some secret string.

Lastly, we conclude the thesis in Chapter 6.

Chapter 2

Cryptanalysis of Simpira v2

2.1 Introduction

In the recent development of the symmetric-key cryptographic algorithms, permutation is one of the commonly used building blocks. Permutations can be used to build block ciphers. A well-known result is to use the (iterated) Even-Mansour construction [59,64] with a permutation to construct a block cipher. Permutations can be used to design hash functions. This kind of application can be found in the SHA-3 finalist Keccak [21], Grøst [68] and JH [138]. It can be used to design authenticated cipher as well. In the recent CAESAR competition, many candidates are based on permutations, such as NORX [12], Ketje [23], Keyak [24], Ascon [55] and ICEPOLE [102,103].

Simpira v2 [71] is a family of permutations proposed by Gueron and Mouha in Asiacrypt 2016. Different from the previous permutations designs, Simpira provides a scalable framework which can process any multiple of the 128-bit AES block size. The Generalized Feistel Structure (GFS) with an F-function that consists of two rounds AES operation is used when the block size is more than 128-bit. The designers also proposed several applications of Simpira v2: a block cipher using Even-Mansour construction, a hash function using Davies-Meyer construction, a wide-block encryption scheme and an authenticated encryption scheme. The security of Simpira is 2^{128} for

distinguishing attacks. The designers also claim that when the permutation is at least 256-bit, the security of Simpira-based block cipher has 128-bit security. The software performance of Simpira with block number $b \leq 4$ and $b = 6$ is below 1 cycle per byte on the Intel Skylake processor using AES-NI.

Previous work. In [123], Rønjom analysed the invariant subspaces on Simpira v1 [70], the first version of Simpira. It was shown in the analysis that invariant subspaces existed in the even-round of Simpira-4, the 4-branch variant of the Simpira family. Dobraunig et. al. [54] independently proposed an attack on Simpira-4 which exploited the property of the GFS of Simpira-4. Due to those analysis, the designers of Simpira have updated the design with a different GFS for Simpira-4 as well as the definition of round constant for all the Simpira family. Zong et al. [149] presented the first cryptanalysis results on Simpira v2. They provided a 9-round impossible differential for Simpira-4 and presented a partial key-recovery attack on the 7-round Simpira-4-based block cipher. The complexity of this attack is 2^{57} to recover 256 bits key. For the 8-round case, the complexity is 2^{170} to recover all the key bits.

Our contributions. Considering the instances in the Simpira family, Simpira-1 is the 12-round AES without round-key; Simpira-2 is an application of the standard Feistel network; Simpira-4 has been studied in the previous analysis. But the properties of Simpira-3 have not been thoughtfully analysed in any previous work. In this chapter, we give the first analysis on the Simpira-3. The summary of our results are listed below.

- Truncated Differential Analysis:

- 8 rounds:

- Distinguishing attack with 2 queries, 0 probability of false negative and 2^{-96} probability of false positive

- 9 rounds:

- Distinguishing attack with 2^{22} inverse queries, 0.0001 probability of false negative and 0.00003 probability of false positive.
 - Key Recovery attack with 2^{24} decryption queries to recover k_0 and k_2
- Impossible Differential Analysis:

9 rounds:

- Key Recovery attack with 2^{50} decryption queries to recover k_0 .

10 rounds:

- Key Recovery attack with 2^{70} decryption queries to recover k_1 .

- Boomerang attack

10 rounds:

- Distinguishing attack with 2^{23} queries which distinguishes Simpira-3 from random permutation with failure probability less than 0.1%.
- Key Recovery attack with 2^{53} encryption queries and 2^{85} decryption queries to recover k_0 .

Outline. The rest of this chapter is organized as follows. In Section 2.2, we provide the preliminaries for our analysis. In Section 2.3, we present the truncated differential cryptanalysis on Simpira-3. In Section 2.4, we apply the impossible differential attack on Simpira-3. In Section 2.5, we discuss the boomerang attack on Simpira-3. We conclude this chapter in Section 2.7.

2.2 Preliminaries

2.2.1 Notation

$\oplus$	Bitwise XOR
$\mathcal{P}$	Plaintext space
$\mathcal{C}$	Ciphertext space
$\mathcal{K}$	Key space
S	Intermediate state
$S_j^{[h]}[\ell_1, \dots, \ell_t]$	State of round h , subblock j , byte $\ell_1, \ell_2, \dots, \ell_t$.
0	Zero difference
*	Arbitrary difference

2.2.2 Description of Simpira

We describe the specification for Simpira-3, the 3-branch instance of Simpira, which is the target of our analysis. A complete specification on Simpira v2 can be found in [71]. Simpira-3 is a permutation function that follows the type-1 Generalized Feistel structure with 3 branches of size 128 bits each. Figure 2.1 illustrates how Simpira-3 works in three consecutive rounds $3i, 3i+1$ and $3i+2$. Note that we have re-ordered the sub-blocks in rounds $S^{[3i+1]}$ and $S^{[3i+2]}$ for readability. The total number of rounds is 21 for Simpira-3 proposed in [71].

The non-linear function $F_{c,b}$ is a 2 round AES with the first **AddRoundKey** changed to **AddRoundConstant** and the second one omitted. The round constant depends on the value of c, b , where c is a counter that is initialized by one and incremented after every evaluation of $F_{c,b}$. We assign the number of blocks as the value of b , which is 3 for Simpira-3. More specifically:

$$F_{(c,b)}(S) = MC \circ SR \circ SB \circ AC \circ MC \circ SR \circ SB(S).$$

Here MC, SR, SB are **MixColumn**, **ShiftRow** and **SubByte** of the AES [50] respec-

Figure 2.1: Round $3i, 3i + 1, 3i + 2$ of Simpira-3, $0 \leq i \leq 6$.

tively. AC is **AddRoundConstant**.

We will omit the index (c, b) of $F_{(c,b)}$ for the remainder of this chapter.

2.2.3 Even-Mansour Construction

In [64], Even and Mansour proposed a way to use a keyless permutation to build a block cipher. Given a permutation π , the block cipher built by the Even-Mansour construction with a single key K is defined to be $E : \mathcal{P} \times \mathcal{K} \rightarrow \mathcal{C}$ such that for any plaintext $P \in \mathcal{P}$ and key $K \in \mathcal{K}$,

$$E(P, K) = \pi(P \oplus K) \oplus K.$$

In this chapter, we consider the block cipher built from the Even-Mansour construction with the permutation function described in Subsection 2.2.2. So the length of P, K and C are all 384 bits each. The security claim given in [71] is the least between 2^{128} and $2^{n/2}$ where $n = 384$. Hence the security claim is 2^{128} while it does not claim any security for the attack with bigger complexity. So we aim for attacks with complexity less than 2^{128} .

2.3 Truncated Differential Attack on Simpira-3

In this subsection, we review some basic concepts and properties of Simpira-3. Then we apply the truncated differential cryptanalysis [84] to study the security of Simpira-3.

First, we recall the definition of Super S-Box as defined in [51].

Definition 2.3.1. [51] The AES Super Sbox SS maps a 4-byte array (a_0, a_1, a_2, a_3) to a 4-byte array (b_0, b_1, b_2, b_3) using a 4-byte array key (k_0, k_1, k_2, k_3) . It consists of the sequence of four transformations: **SubByte**, **MixColumn**, **AddRoundKey** and **SubByte**.

By this definition and the commutativity of **SubByte** and **ShiftRow**, we can rewrite the definition of $F(S)$ as

$$F(S) = MC \circ SR \circ SS \circ SR(S).$$

The Super SBox has the following property:

Property 2.3.2. Fix a key $K = (k_0, k_1, k_2, k_3) \in \mathbb{F}_2^{32}$ and consider non-zero input and output differences $\Delta input, \Delta output \in \mathbb{F}_2^{32}$. Then in average, the equation:

$$SS(x) \oplus SS(x \oplus \Delta input) = \Delta output$$

has one solution for x .

In our analysis, we consider two types of byte difference, either 0 (must be zero) or $*$ (arbitrary value). Let $\Delta S_1, \Delta S_2$ be two 4×4 non-zero matrices with elements from $\{0, *\}$ representing two truncated differential pattern in some state. We say $\Delta S_1 \neq \Delta S_2$ if

$$\{(i, j) | 0 \leq i, j \leq 3, \Delta S_1(i, j) = *\} \cap \{(i, j) | 0 \leq i, j \leq 3, \Delta S_2(i, j) = *\} = \emptyset.$$

The following observation is made in [149].

Observation 2.3.3. *If there exists at least one column of $SR \circ SB(\Delta S)$ to be inactive, then the number of possible values of $\Delta F(S)$ is strictly less than 2^{128} .*

This means that if $SR \circ SB(\Delta S)$ has at least one zero columns, $\Delta F(S)$ will have some pattern that we can recognize.

For the Generalized Feistel structure of 3 branches, there exists asymmetry between the forward and backward direction. More specifically, in the forward direction, any difference introduced through F will pass through F in each of the following rounds, this is not true for the backward direction. Every time a difference is introduced to a branch by F , it will pass through one round untouched before having to go through another F . This leads to a slower diffusion in the backward direction than the forward direction. In our truncated differential cryptanalysis, we will exploit this property to study the inverse permutation.

After searching the possible truncated differential of Simpira-3, the following observation gives the best 8-round truncated differential that we found.

Observation 2.3.4. *(8-Round Backward Truncated Differential) Let a be a truncated differential pattern such that $SR \circ SB(a)$ has at least one zero column. Then the differential*

$$(0, 0, a) \xrightarrow{8R \text{ backwards}} (F(a), r, r)$$

has probability 1. This means that if $\Delta S^{[8]} = (0, 0, a)$, then with probability 1, we have $\Delta S^{[0]} = (F(a), r, r)$. Here we use r to represent the unknown difference in some sub-blocks. For example $F(F(a))$ is not considered as identifiable, so we label it as r .

Proof. It is easy to see that the following 8-round backwards differential holds with probability 1 :

$$\begin{aligned} (0, 0, a) &\rightarrow (0, 0, a) \rightarrow (0, 0, a) \rightarrow (F(a), 0, a) \rightarrow (F(a), 0, a) \\ &\rightarrow (F(a), r, a) \rightarrow (F(a), r, a) \rightarrow (F(a), r, r) \rightarrow (F(a), r, r). \end{aligned}$$

This characteristic is depicted in Figure 2.2.

□

Figure 2.2: Backward Difference Propagation of the 8-round Simpira-3 $(0, 0, a) \xrightarrow{8R} (F(a), r, r)$.

2.3.1 Distinguishing Attacks on Simpira-3

In this subsection, we describe the distinguishing attack on Simpira-3 based on the truncated differential characteristic we found above. We propose two different distinguishing attacks, one for the 8-round Simpira-3 and another for the 9-round Simpira-3.

2.3.1.1 8-Round Distinguishing Attack

Suppose that Π is a permutation function defined as $\Pi : (\mathbb{F}_2^{128})^3 \rightarrow (\mathbb{F}_2^{128})^3$. We will identify if Π is an 8-round Simpira-3 or a random permutation using the following procedure.

1. Select a differential pattern a such that $SR \circ SB(a)$ has three zero columns. Possible values of a are depicted in Figure 2.3. Choose $C_1, C_2 \in (\mathbb{F}_2^{128})^3$ such that $C_1 \oplus C_2 \in (0, 0, a)$.

Figure 2.3: Four possible pattern of a . Filled cells denote arbitrary differences as long as they are not all zero.

Figure 2.4: Respective patterns of $MC^{-1}(F(a))$.

2. Input C_1 and C_2 to the inverse oracle of Π to obtain $P = (p_0^1, p_1^1, p_2^1)$ and $P_2 = (p_0^2, p_1^2, p_2^2)$.
3. Calculate $MC^{-1}(p_0^1 \oplus p_0^2)$. If the pattern does not follow the corresponding pattern depicted in Figure 2.4, we conclude that Π is a random permutation. Otherwise, we conclude that it is Simpira-3.

Note that if Π is Simpira-3, by Observation 2.3.4, given a difference of pattern $(0, 0, a)$, with probability 1 after 8 inverse rounds, the pattern will be $(F(a), r, r)$. So the probability of a false negative in our attack is 0. On the other hand, if Π is a random permutation, $p_0^1 \oplus p_0^2$ is supposed to be a random 128-bit state. Hence $MC^{-1}(p_0^1 \oplus p_0^2)$ is again a random 128-bit state. The probability that it has the corresponding pattern given in Figure 2.4 is at most the probability that the value in the 12 blank cells are all zero, which is $(2^{-8})^{12} = 2^{-96}$. So the probability of our attack giving a false positive is 2^{-96} . Considering the complexity of the attack, we only require 2 inverse queries and negligible processing time and storage.

2.3.1.2 9-Round Distinguishing Attack

In this attack, we will add a round after the 8-round truncated differential characteristic. Namely $(F(a), 0, a) \rightarrow (0, 0, a)$, which is depicted in Figure 2.5.

Figure 2.5: Round 9 in Distinguishing Attack.

The 9-round truncated differential distinguisher can be done in the following way. We are assuming that we are given a permutation $\Pi : (\mathbb{F}_2^{128})^3 \rightarrow (\mathbb{F}_2^{128})^3$ and we are asked to identify whether Π is a 9-round Simpira-3 or a random function.

1. Fix the pattern a as the last pattern given in Figure 2.3 and $F(a)$ as the MixColumn of its corresponding pattern in Figure 2.4. In other words, now the non-zero cells of a are in cells 1, 6, 11, 12 following the numbering of state cells in [50]. Similarly, the possible non-zero cells of $MC^{-1}(F(a))$ are in cells 3, 6, 9, 12.
2. Fix $c_1 \in \mathbb{F}_2^{128}$, $c_0[0, 1, 2, 4, 5, 6, 8, 10, 11, 13, 14, 15]$ and $c_2[0, 1, \dots, 11, 13, 14, 15]$.
3. For $i, j = 0, \dots, 2^8 - 1$, we define $c_0^{(2^8 i + j)}[3, 6, 9, 12] = (i, j, 0, 0)$ and $c_0^{(2^{16} + 2^8 i + j)}[3, 6, 9, 12] = (0, 0, i, j)$. So now we have $c_0^{(i)}$ for $i = 0, \dots, 2^{17} - 1$.
4. For $i = 0, \dots, 2^4 - 1$, we define $c_2^{(i)}[12] = i$ and $c_2^{(2^4 + i)}[12] = 2^4 + i$. So now we have $c_2^{(j)}$ for $j = 0, \dots, 2^5 - 1$.
5. For $i = 0, \dots, 2^{17} - 1, j = 0, \dots, 2^5 - 1$, calculate

$$\Pi^{-1}(c_0^{(i)}, c_1, c_2^{(j)}) = (p_0^{(i,j)}, p_1^{(i,j)}, p_2^{(i,j)}).$$

From these plaintexts, we can generate 2^{43} plaintext pairs.

6. Count the number of plaintext pairs that have difference pattern $(F(a), r, r)$.

Let this number be n . If $n \geq 1880$, conclude that Π is Simpira-3. Otherwise we conclude that Π is a random permutation.

Note that if Π is Simpira-3 and $F(\Delta c_2) = \Delta c_0$, then $\Delta p_0 = F(a)$ with probability 1. So

$$Pr(\Delta p_0 = F(a) | \Pi \text{ is Simpira-3 and } F(\Delta c_2) = \Delta c_0) = Pr(F(\Delta c_2) = \Delta c_0).$$

Now we note that by the choice of our inputs,

$$\{MC^{-1}(\Delta c_0^i)\} = \{\Delta \in \{0, *\}^{4 \times 4}, \Delta[0, 1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 15] = 0\}.$$

Moreover,

$$\{\Delta c_2^j\} = \{\Delta \in \{0, *\}^{4 \times 4}, \Delta[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13, 14, 15] = 0\}.$$

By design, $MC^{-1}(F(\Delta c_2^j))$ has the fourth pattern given in Figure 2.4. So the probability above equals to the probability that $MC^{-1}(\Delta c_0^i)$ coincides with the value we get from $MC^{-1}(F(\Delta c_2^j))$ which only depends on the value of the 4 values in cells 3, 6, 9, 12. Hence this probability is 2^{-32} .

Furthermore, if Π is Simpira-3, $F(\Delta c_2) \neq \Delta c_0$ or if Π is a random permutation, Δp_0 can be seen as a random 128-bit state. So we also have that $MC^{-1}(\Delta p_0)$ is also a random 128-bit state. So for this value to be in the $MC^{-1}(F(a))$ pattern, the probability, as discussed before, is 2^{-96} . So now we have that if Π is Simpira-3, then the number of pairs that has Δp_0 in $F(a)$ pattern follows a binomial distribution with 2^{43} trials and $2^{-32} + 2^{-96}(1 - 2^{-32}) \approx 2^{-32}$ success probability. Let this random variable be denoted by X_1 . On the other hand, if Π is a random permutation, then this random variable follows another binomial distribution with 2^{43} trials and 2^{-96} success probability. Let this random variable be denoted by X_2 . Then our attack's probability of false negative is

$$\alpha = Pr(n \leq 1879 | n \sim X_1).$$

and our attack's false positive probability is

$$\beta = \Pr(n \geq 1880 | n \sim X_2).$$

A simple calculation using the normal approximation of random variables will tell us that $\alpha \approx 0.0001$ and $\beta \approx 0.00003$.

Lastly, for the complexity, our attack complexity is 2^{22} inverse enquiries and 2^{43} comparison.

2.3.1.3 9-Round Key Recovery Using Truncated Differential

The distinguishing attacks given above can easily be adapted to attack the block cipher constructed by using the Even-Mansour construction with Simpira-3.

Assuming that the encryption scheme follows the (single key) Even-Mansour structure with key $K = (k_0, k_1, k_2)$ and the permutation function being the 9-round Simpira-3, we can use the 9-round truncated differential distinguisher to launch the partial key recovery attack.

1. Select a according to the first pattern given in Figure 2.3. $F(a)$ is the difference after MixColumn of the corresponding pattern in Figure 2.4.
2. Generate $(c_0^{(i)}, c^1, c_2^{(j)})$ with an analogous method as the one we described in the 9-rounds distinguishing attack in Subsection 2.3.1.2.
3. Decrypt $(c_0^{(i)}, c^1, c_2^{(j)})$ to obtain 2^{22} plaintexts $(p_0^{(i,j)}, p_1^{(i,j)}, p_2^{(i,j)})$ and generate 2^{43} plaintext-ciphertext pairs from it, all with the ciphertext differences in the pattern $(F(a), 0, a)$.
4. Out of the 2^{43} pairs, we have a probability 2^{-32} for the corresponding Δp_0 to have difference pattern $F(a)$. When this happens with high probability we have $F(\Delta c_2) = \Delta c_0$. We can expect around 2^{11} pairs that satisfy this. Let the differences in the first and last sub-blocks be $(\Delta c_0^{(i)}, \Delta c_2^{(i)}), i = 1, \dots, 2^{11}$ with its corresponding ciphertexts being $(c_0^{(i,1)}, c_2^{(i,1)})$ and $(c_0^{(i,2)}, c_2^{(i,2)})$. Due to the

linearity of SR and MC , we also have that $SS(SR(\Delta c_2)) = MC^{-1} \circ SR^{-1}(\Delta c_0)$. By Observation 2.3.2, we can identify the value of $(c_0^{(i,1)} \oplus k_0, c_2^{(i,1)} \oplus k_2)$ and $(c_0^{(i,2)} \oplus k_0, c_2^{(i,2)} \oplus k_2)$. Note that since the observation only mentions that there is in average one pair of solutions for $(c_0^{(i,1)} \oplus k_0, c_2^{(i,1)} \oplus k_2)$ and $(c_0^{(i,2)} \oplus k_0, c_2^{(i,2)} \oplus k_2)$, there are two possibilities of matching the pair. So for each of the filtered $(\Delta c_0, \Delta c_2)$, we obtain two possible values of $(k_0[0, 5, 10, 15], k_2[0, 7, 10, 13])$. Note that the two possible values that we obtain from $(\Delta c_0^{(i)}, \Delta c_2^{(i)})$ will contain a correct key

$$(k_0^*[0, 5, 10, 15], k_2^*[0, 7, 10, 13])$$

and

$$(k_0^*[0, 5, 10, 15] \oplus \Delta_0, k_2^*[0, 7, 10, 13] \oplus \Delta_2)$$

where $\Delta_0 = c_0^{(i,1)} \oplus c_0^{(i,2)}$ and $\Delta_2 = c_2^{(i,1)} \oplus c_2^{(i,2)}$.

So a wrong key can be in at least two pairs if the difference in both the first and last sub-blocks in the two pairs are the same. Now a specific difference in the last block can only occur in at most 2^7 different pairs by our choices of c_2 . Since we have 2^{11} pairs, we can expect to filter out all the wrong $(k_0[0, 5, 10, 15], k_2[0, 7, 10, 13])$.

5. Repeating the whole process with a as the second, third and last patterns given in Figure 2.3 will help us in recovering the other 12 bytes of k_0 and k_2 .

Note that here the complexity of each iteration is 2^{22} decryption enquiry and 2^{43} processing of the pairs and there are 4 iterations. So in total, the time complexity is 2^{24} to recover the value of k_0 and k_2 .

2.4 Impossible Differential Attack on Simpira-3

In this section, we will use the impossible differential [25, 83] to analyse the block cipher based on Simpira-3. We remark that the analysis described in this section can

be adapted to develop distinguishing attack using a similar method.

The backward truncated differential trail we found in the previous section can be extended to an impossible differential trail in the following observation.

Observation 2.4.1. (*9-round Impossible Differential*) *Let a and b be two truncated differential patterns such that $SR \circ SB(a)$ and $SR \circ SB(b)$ have at least one zero column and the nonzeros of $SR \circ SB(a)$ and $SR \circ SB(b)$ lie in different column. Then the differential:*

$$(b, F(b), r) \xrightarrow{9R} (a, 0, 0)$$

is impossible.

Proof. We can see that the following 1 round forward truncated differential has probability 1.

$$(b, F(b), r) \rightarrow (b, F(b), r).$$

In the backward direction, as discussed in Observation 2.3.4, the differential pattern follows the following path:

$$\begin{aligned} (a, 0, 0) &\rightarrow (a, 0, 0) \rightarrow (a, 0, 0) \rightarrow (a, F(a), 0) \rightarrow (a, F(a), 0) \\ &\rightarrow (a, F(a), r) \rightarrow (a, F(a), r) \rightarrow (r, F(a), r) \rightarrow (r, F(a), r). \end{aligned}$$

If the trail is valid, we must have some difference that is in both patterns, $F(b)$ and $F(a)$. Since the nonzero columns of $SR \circ SB(a)$ is completely different from the nonzero columns of $SR \circ SB(b)$, we have $MC^{-1}(F(a)) \neq MC^{-1}(F(b))$. But it is impossible for a non-zero difference to be both in $F(a)$ and $F(b)$. Hence this 9 rounds differential path is an impossible differential. $\square$

For the following attacks, we will consider two families of patterns which are denoted by a and b . The first family, a , is already defined in Figures 2.3 and 2.4. The other family, b , is in fact the complement of a , depicted in Figures 2.6 and 2.7 respectively.

Figure 2.6: Four possible patterns of b .

Figure 2.7: Patterns of $MC^{-1}(F(b))$ corresponding to Figure 2.6.

2.4.1 Key Recovery Attack on 10-Round Simpira-3

For the key recovery attack for E_K with Π being the 10-round Simpira-3, we use the 9-round impossible differential we described in Observation 2.4.1 and then append it with the rotated variant of the round given in Figure 2.5 which is $(a, 0, 0) \xrightarrow{1R} (a, F(a), 0)$.

We describe the 10-round attack in the following steps:

1. Let a be the first pattern given in Figure 2.3. We also fix b to be the corresponding pattern in Figure 2.6. Lastly, let $F(a)$ and $F(b)$ be the `MixColumn` of the corresponding patterns in Figures 2.4 and 2.7 respectively.
2. Construct 2^{34} ciphertexts differing only in the active cells of $(a, F(a), 0)$. This gives us 2^{67} ciphertext pairs each with difference pattern $(a, F(a), 0)$.
3. Decrypt the 2^{34} ciphertexts to find the corresponding plaintexts. Filter the pairs with plaintext difference $(b, F(b), r)$. This gives us a 64 bit filter. Therefore, we expect to have 2^3 pairs out of the 2^{67} pairs of plaintext with differences pattern $(b, F(b), r)$.
4. For each of the remaining pair, let the ciphertext difference be $(\Delta c_0, \Delta c_1, \Delta c_2)$ and the plaintext difference be $(\Delta p_0, \Delta p_1, \Delta p_2)$. From the impossible differential

in Observation 2.4.1, the relation $(a, 0, 0) \rightarrow (a, F(a), 0)$ in Round 10 must be invalid. This gives a filter of one value of the key bytes $k_1[0, 7, 10, 13]$ (See Step 4 in Subsection 2.3.1.3 for more details).

5. Repeat this 2^{34} times to filter out all wrong $k_1[0, 7, 10, 13]$.
6. Repeat the whole process for other patterns of $(a, F(a), b, F(b))$ to recover the remaining bytes of k_1 .

The reasoning behind the 2^{34} repetition is exactly the same as the reasoning of the repetition done in the 9-round key recovery using truncated differential found in Subsection 2.3.1.3. The complexity of this attack is $2^{34} \times 2^{34} \times 4 \approx 2^{70}$ decryption queries and 2^{104} pairs to recover k_1 .

We can actually also mount a key recovery attack on 9-round Simpira-3, which can be found in the following subsection.

2.4.2 Key Recovery Attack on 9-Round Simpira-3

We define E_K to be the block cipher built by using the Even-Mansour construction with 9-round Simpira-3.

For this key recovery attack, we use a shortened variant of the 9-round impossible differential that we have in Observation 2.4.1 where we omit the first round of that trail. Therefore, we need to rotate the whole trail and it changes the input difference pattern from $(F(b), r, b)$ to $(F(b), r, r)$. This 8-round impossible differential can be found in the Figure 2.8.

Based on this differential, we then append one more round, which is the round given in Figure 2.5. The attack procedure is given as follows:

1. Let a be the first pattern in Figure 2.3, and b be the first pattern given in Figure 2.6, which is the complementary of a . Let $F(a)$ and $F(b)$ be the `MixColumn` of the corresponding patterns in Figure 2.4 and Figure 2.7 respectively.

Figure 2.8: 8-round impossible differential used for 9-round impossible distinguisher

2. Construct 2^{22} ciphertexts that differ in the active cells of $(F(a), 0, a)$. This provides us with 2^{43} ciphertext pairs with difference pattern $(F(a), 0, a)$.
3. Decrypt the 2^{22} ciphertexts to get 2^{22} plaintexts. For each of the 2^{43} pairs of plaintext, we check if they have difference pattern $(F(b), r, r)$. Since $F(b)$ provides a 32-bit filter, we expect to have 2^{11} pairs of plaintext with difference $(F(b), r, r)$.
4. For each of the remaining pair, let the ciphertext difference be $(\Delta c_0, \Delta c_1, \Delta c_2)$ and the plaintext difference be $(\Delta p_0, \Delta p_1, \Delta p_2)$. From the impossible differential in Figure 2.8, the relation $(0, 0, a) \rightarrow (F(a), 0, a)$ in Round 9 must be invalid. This gives a filter of one value of the key bytes $k_0[0, 7, 10, 13]$ (See Step 4 in Subsection 2.3.1.3 for more details).
5. Repeat the above procedure for 2^{26} times to detect all wrong possible values of $k_0[0, 7, 10, 13]$.
6. Repeat the whole process for the other 3 patterns of $(a, F(a), b, F(b))$ to recover

the remaining bytes of k_0 .

For each iteration, we are left with 2^{11} pairs. With 2^{26} iterations, the total number of pairs is 2^{37} . The probability that any wrong key left after the attack is $(1 - 2^{-32})^{2^{37}} < 2^{-32}$. Hence we expect to have only one key remaining after 2^{26} iterations.

In total we have $2^{22} \times 2^{26} \times 4 \approx 2^{50}$ decryption queries and 2^{51} pairs to process to recover the value of k_0 .

2.5 Boomerang Attack on Simpira-3

The boomerang attack, proposed by David Wagner in FSE 1999 [132] is a powerful tool to analyse block ciphers and permutations. To apply the boomerang attack, we need to concatenate two short differential trails to make a longer one.

In our analysis, we apply it to the truncated differential of Simpira-3. We first restate the following result from [132] in finding the success probability of the boomerang attack.

Lemma 2.5.1. *[132] Let E be an encryption operation that can be decomposed to $E = E_1 \circ E_0$. We further let $\Delta \rightarrow \Delta^*$ be a truncated differential characteristic for E_0 and $\nabla \rightarrow \nabla^*$ be a truncated differential characteristic for E_1^{-1} . Then a truncated boomerang attack using these two characteristics has success probability p where:*

$$p \approx Pr(\Delta \rightarrow \Delta^*) \times Pr(\nabla \rightarrow \nabla^*)^2 \times Pr(\Delta^* \rightarrow \Delta) \\ \times Pr(w \oplus x \oplus y \in \Delta^* | w \in \Delta^*, x, y \in \nabla^*).$$

A slight modification on Observation 2.4.1 gives us the following observation:

Observation 2.5.2. *Let b be one of the pattern given in Figure 2.6. Then the truncated boomerang attack for 9 rounds with:*

- $\Delta = (b, F(b), r) \xrightarrow{1R} \Delta^* = (b, F(b), r)$

- $\nabla = (b, 0, 0) \xrightarrow{8R} \nabla^* = (r, F(b), r)$

has success probability 2^{-32} .

Proof. By Observation 2.4.1 we have that $Pr(\Delta \rightarrow \Delta^*) = Pr(\nabla \rightarrow \nabla^*) = 1$. Furthermore, it is easy to see that $Pr(\Delta^* \rightarrow \Delta) = 1$. So the success probability p of this attack is the same as the probability that $\mathbf{w} = (w_0, w_1, w_2) \oplus \mathbf{x} = (x_0, x_1, x_2) \oplus \mathbf{y} = (y_0, y_1, y_2) \in (b, F(b), r)$ given that $\mathbf{w} \in (b, F(b), r), \mathbf{x}, \mathbf{y} \in (r, F(b), r)$. Now note that this probability is one for the second and third branches. So we have the success probability, p is:

$$\begin{aligned}
p &\approx Pr(\mathbf{w} \oplus \mathbf{x} \oplus \mathbf{y} \in (b, F(b), r) | \mathbf{w} \in (b, F(b), r), \mathbf{x}, \mathbf{y} \in (r, F(b), r)) \\
&= Pr(w_0 \oplus x_0 \oplus y_0 \in b | w_0 \in b, x_0, y_0 \in r) \\
&= Pr(z_0 \in b | z_0 \in r) \\
&= Pr(z_0 \text{ has 4 zeros in the four specific cells}) \\
&= (2^{-8})^4 = 2^{-32}.
\end{aligned}$$

□

2.5.1 Distinguishing Attack on 10- Rounds Simpira-3

The differences used in our boomerang attack are given below:

- $\Delta = (b, F(b), r),$
- $\Delta^* = (b, F(b), r),$
- $\nabla = (b, 0, 0),$
- $\nabla^* = (r, F(b), r),$

where b is one of the four patterns described in Figure 2.6.

Assume that we are given a permutation function Π and we are asked to identify whether Π is a 10-round Simpira-3 or a random permutation. Again we will use the

rotated pattern given in Figure 2.5 such that it follows the form for the 10-th round of Simpira-3.

1. Fix a pattern b to be one of the pattern given in Figure 2.3 and $F(b)$ be the **MixColumn** of the corresponding pattern given in Figure 2.7.
2. Generate 2^{22} different $P^{(i)}, i = 1, \dots, 2^{22}$ such that for any $i, j, P^{(i)} \oplus P^{(j)} \in (b, F(b), r)$.
3. Compute $C^{(i)} = \Pi(P^{(i)})$.
4. Generate $d_0^{(i)}$ such that $c_0^{(i)} \oplus d_0^{(i)} \in b$.
5. Compute $\delta = F(c_0^{(i)}) \oplus F(d_0^{(i)})$ and generate $d_1^{(i)} = c_1^{(i)} \oplus \delta$. Lastly, we let $d_2^{(i)} = c_2^{(i)}$. Define $D^{(i)} = (d_0^{(i)}, d_1^{(i)}, d_2^{(i)})$.
6. Compute $Q^{(i)} = \Pi^{-1}(D^{(i)})$.
7. Compute $n = |\{\{i, j\} : i \neq j, Q^{(i)} \oplus Q^{(j)} \in (b, F(b), r)\}|$.
8. If $n \geq 1880$, conclude that Π is Simpira-3, otherwise, conclude that Π is a random permutation.

Note that if Π is Simpira, the probability that $Q^{(i)} \oplus Q^{(j)} \in (b, F(b), r)$ is 2^{-32} by Observation 2.5.2. On the other hand, this probability is 2^{-64} when Π is a random permutation. With a similar argument in the 9-round truncated differential distinguishing attack, the probabilities of false negative and false positive are 0.0001 and 0.00003 respectively. The complexity of this attack is $2^{22}\Pi$ queries, $2^{22}\Pi^{-1}$ queries and 2^{43} pairs to process.

Here we need to use the actual input value of the state in the last round's F . Hence this distinguisher of the permutation function cannot be applied to distinguish the block cipher using Even-Mansour construction with 10-round Simpira-3 as its permutation function.

Moreover, we can construct a distinguisher for 9-round Simpira-3 and launch a partial key recovery attack for 10-round Simpira-3 based on it. The details are provided in the following subsections.

2.5.2 Distinguishing Attack on 9- Rounds Simpira-3

In this case, we will be using the following values:

- $\Delta = (a, F(a), r)$,
- $\Delta^* = (a, F(a), r)$,
- $\nabla = (a, 0, 0)$ and
- $\nabla^* = (r, F(a), r)$

where a is one of the patterns described in Figure 2.3.

By the same argument as the one given in Observation 2.5.2, it is easy to see that the success probability of this boomerang trail is 2^{-96} .

Assume that we are given a permutation function Π and we are asked to identify whether Π is 9-round Simpira-3 or a random permutation.

1. Fix a pattern a to be one of the pattern given in Figure 2.3 and $F(a)$ be `MixColumn` of the corresponding pattern given in Figure 2.7.
2. Generate $2^{51}P^{(i)}, i = 1, \dots, 2^{51}$ such that for any $i, j, P^{(i)} \oplus P^{(j)} \in (a, F(a), r)$.
3. Calculate $C^{(i)} = \Pi(P^{(i)})$.
4. Generate $D^{(i)}$ such that $C^{(i)} \oplus D^{(i)} \in (a, 0, 0)$
5. Calculate $Q^{(i)} = \Pi^{-1}(D^{(i)})$.
6. Compute $n = |\{(i, j) | i \neq j, Q^{(i)} \oplus Q^{(j)} \in (a, F(a), r)\}|$.

7. If $n \geq 2$, conclude that Π is Simpira-3, otherwise, conclude that Π is a random permutation.

We note that in this attack, if Π is Simpira, the probability that $Q^{(i)} \oplus Q^{(j)} \in (a, F(a), r)$ is 2^{-96} by the observation we made above. This probability becomes 2^{-192} when Π is a random permutation. So by a similar argument in the 9-Rounds Distinguishing attack based on Truncated differential analysis, the probability of false negative and false positive are 0.0001 and 0.00003 respectively. The complexity here is $2^{51}\Pi$ queries, $2^{51}\Pi^{-1}$ queries and 2^{100} pairs to process.

The distinguisher we introduced above in fact can be improved if we are using b instead of a . However, we will be using the distinguisher described here to launch a key recovery attack on 10-rounds Simpira-3 and the complexity of the key recovery attack is the smallest if we use a .

2.5.3 10-Round Key Recovery Attack Using Boomerang

As before, let E_K be a block cipher built by using the Even Mansour scheme with 10-round Simpira-3 as its permutation function. For this attack, we will call the active diagonal in each of the pattern given in Figure 2.3 as first, second, third and fourth diagonal respectively. So for example, in the first pattern given in Figure 2.6, the active diagonals are second, third and fourth diagonal. Similarly, we call the active anti-diagonal in each of the pattern given in Figure 2.4 as the first, second, third and fourth anti-diagonal respectively.

The attack goes as follows:

1. Let a be the first pattern given in Figure 2.3 and $F(a)$ be the MixColumn of the corresponding pattern in Figure 2.4.
2. Construct 2^{51} plaintexts $P^{(i)}, i = 1, \dots, 2^{51}$ such that for any i and j ,

$$\forall i, j, P^{(i)} \oplus P^{(j)} \in (a, F(a), r).$$

3. Calculate $C^{(i)} = (c_0^{(i)}, c_1^{(i)}, c_2^{(i)}) = E_K(P^{(i)})$.
4. For each guess of the value of the first diagonal of k_0 , generate $d_0^{(i)}$ such that $d_0^{(i)} \oplus c_0^{(i)} \in a$. Since we guessed the value of the keys in the active cells, we know the exact value of $\delta = F(d_0^{(i)} \oplus c_0^{(i)})$. Generate $d_1^{(i)}$ such that $d_0^{(i)} \oplus c_0^{(i)} = \delta$. Set $d_2^{(i)} = c_2^{(i)}$ and let $D^{(i)} = (d_0^{(i)}, d_1^{(i)}, d_2^{(i)})$.
5. Calculate $Q^{(i)} = E_K^{-1}(D^{(i)})$.
6. Compute $n = |\{(i, j) | i \neq j, Q^{(i)} \oplus Q^{(j)} \in (a, F(a), r)\}|$. If $n \geq 2$, conclude that the current guess of $k_0[0, 5, 10, 15]$ passes the filter. Otherwise, conclude the current guess is wrong repeat with different guess of the 4-byte subkey.
7. Repeat the whole attack with the the other three patterns of a to retrieve the value of k_0 in the other three diagonals.

Note that in this attack, we assume that if our guess key is wrong, then the remaining 9 rounds is a random permutation instead of 9-round Simpira-3 that is discussed in the 9-round distinguishing attack. So we use the 9-round distinguishing attack here to decide whether the current guess of key is correct or not. So with failure probability of less than 0.1%, we can recover the value of k_0 . For the complexity, for each pattern of a , we recover one diagonal of k_0 with complexity 2^{51} encryption queries and $2^{32} \times 2^{51} = 2^{83}$ decryption queries. Since we repeat the attack four times for four different values of a , the total complexity is 2^{53} encryption queries and 2^{85} decryption queries.

We note here that this attack recovers different subkey from the one recovered in the key recovery attack of 10-round Simpira-3 based on impossible differential attack. The attack described in this subsection recovers the value of k_0 while the previous one recovers the value of k_1 .

2.6 Summary of Findings

The following table summarizes all the attacks we have proposed in the previous sections. In this table, complexity of $2^{53} + 2^{85}$ means 2^{53} encryption queries and 2^{85} decryption queries. Furthermore, for the key recovery attack, the format will be $a; b$ where a represents the complexity and b represents the part of the key that we can recover.

Table 2.1: Table of Complexity of Distinguishing and Key Recovery Attacks using Truncated Differential, Impossible Differential and Boomerang Cryptanalysis in Different Number of Rounds

Number of Query	Distinguishing Attack			Key Recovery Attack	
	8	9	10	9	10
Truncated Differential Attack	2	2^{22}	N.A.	$2^{24}; k_0, k_2$	N.A.
Impossible Differential Attack	N.A.	N.A.	N.A.	$2^{50}; k_0$	$2^{70}; k_1$
Boomerang Attack	N.A.	N.A.	2^{23}	N.A.	$2^{53} + 2^{85}; k_0$.

2.7 Conclusion

In this section, we described three types of cryptanalysis on Simpira-3, namely truncated differential analysis, impossible differential analysis and boomerang attack. Applying the cryptanalysis techniques, we present the first practical distinguisher for 9-round and 10-round Simpira-3 permutations with complexity 2^{22} and 2^{23} respectively. We also present partial key recovery attacks on the 9-round and 10-round Simpira-3 based block ciphers, which greatly reduce the key space in those instances. We remark that our attacks on Simpira-3 cannot directly extend to Simpira-4 to improve the previous results, as Simpira-4 has a more symmetric structure.

Our attacks provide the first concrete security analysis on Simpira-3. The results

confirm that Simpira-3 has a good security margin and we hope that our results will be useful for the future analysis on the Simpira family of permutations.

Chapter 3

Improved Differential Cryptanalysis on Generalized Feistel Schemes

3.1 Introduction

3.1.1 Background

The Feistel Network is a widely-used method used to construct iterated block ciphers. It has similar operations in encryption and decryption process which is hardware efficient and the round function is not required to be a bijective function. It has been applied in many block ciphers such as DES and Camellia [10]. The structure was generalized to allow more branches and different relations between the branches to form Generalized Feistel Network (GFN) [111], which was again applied in many block cipher designs such as CLEFIA [2] and Simpira [71]. The advantage of using a Generalized Feistel Network is that it allows for a design to handle a larger block size with a relatively small round function.

Many analysis on Feistel network and Generalized Feistel Network have been done [75, 95, 105, 113, 114, 130]. However, as mentioned in [75], most analysis is specialized in some types instead of analysing many types at once.

In [105], Nachev et.al. used differential cryptanalysis to study four types of GFS using Known Plaintext Attack (KPA) and Chosen Plaintext Attack (CPA) model. They established lower bounds of the maximum number of rounds distinguishable in Type-1, Type-2, Type-3 and Alternating Feistel Scheme in the two models.

Provable-security analysis has been applied to Feistel Networks in [95] and [75]. In [95], Luby and Rackoff analysed the classical Feistel Networks which is then improved and generalised by Hoang and Rogaway in [75] to analyse Classical, Unbalanced, Alternating, Type-1, Type-2 and Type-3 Feistel Scheme.

3.1.2 Our Contribution

In this chapter, we provide better lower bounds of the maximum number of rounds distinguishable in 3 different types of Generalized Feistel Networks given in [105]. We also provide a lower bound of the maximum number of rounds distinguishable in another type of Generalized Feistel Network, the Unbalanced Feistel Network. As far as we know, this is the first result on Unbalanced Feistel Network that is applicable to different values of k' . We exploit the asymmetry of certain types of GFS by observing that the backward differential diffusion is slower than the forward differential diffusion. This leads to the improvements on the lower bounds.

For Type-1 Feistel Scheme, we provide a chosen ciphertext distinguisher which distinguishes $k - 1$ more rounds than the distinguisher given in [105] with the same complexity. Furthermore, when the number of rounds to distinguish is fixed to $ak - 2$ rounds for some integer a in the range $4 \leq a \leq k - 1$, the distinguisher in this chapter has complexity $1/2^n$ of the distinguisher given in the CPA model in [105], from $\sqrt{2} \cdot 2^{(a-2)n}$ to $\sqrt{2} \cdot 2^{(a-3)n}$.

In Type-3 Feistel Scheme, [105] only provides lower bound for the case when the number of branches is at least 6. We propose a distinguisher which can be used for any number of branches and can distinguish up to $k + 2$ rounds in both KPA and CCA model with complexity $\sqrt{2} \cdot 2^{(k-1)n}$ and $\sqrt{2} \cdot 2^{(k-2)n}$ respectively. When k is at

least 6, in the CCA model, a distinguisher for one more rounds than the one given in [105] is constructed.

In Alternating Feistel Scheme, our analysis shows that lower complexity can be achieved in some special cases. More specifically, when the number of round is odd, the complexity is improved by a factor of $2^{\frac{3n}{2}}$ from the distinguisher proposed in [105].

The same analysis is applied to Unbalanced Feistel Scheme with parameter k' and k ($\text{UFN}(k', k)$). In our analysis of Unbalanced Feistel Scheme, k represents the number of sub-blocks in the Generalized Feistel Scheme while k' represents the number of sub-blocks that are used as the output of the round function. In this chapter, we consider two special cases when k' or $k - k'$ divides k . When $k' = 1$, we can distinguish up to $(k^2 + k - 1)$ rounds with complexity less than 2^{kn} under KPA model. In the CCA model, the number of rounds that can be distinguished is up to $2k$ rounds with complexity less than 2^n . When $k' \geq 1$, a lower bound of the maximum number of rounds that is distinguishable from random permutation is given. In KPA model, the bound is $\frac{k^2}{k'} - \frac{k}{2} + \frac{k}{k'}$ when k' is even and $\frac{k^2}{k'} - \frac{k(k-1)}{2k'}$ when k' is odd. In CCA model, the bound is $\frac{k}{2} + 2\frac{k}{k'}$ when k' is even and $\frac{k(k'+3)}{2k'}$ when k' is odd. To the best of our knowledge, this is the first analysis on Unbalanced Feistel Scheme for any values of k .

3.1.3 Organization

Some preliminaries are given in Section 3.2. The attack overview is then discussed in Section 3.4. The analysis on Type-1 Feistel Scheme is presented in Section 3.5. We remark that an analysis of Type-2 Feistel Scheme has been done but the result is not an improvement from the analysis done in [105] and hence omitted. Sections 3.6 and 3.7 contains analysis of Type-3 and Alternating Feistel Scheme and lastly, the Unbalanced Feistel Scheme is analysed in Section 3.8.

3.2 Preliminaries

3.2.1 Generalized Feistel Schemes

A Generalized Feistel Scheme of branch k is defined as a (keyed)-permutation $\Pi : (\mathbb{F}_{2^n})^k \rightarrow (\mathbb{F}_{2^n})^k$. For the m input-output pairs of Π , for all $i \in \{0, \dots, m-1\}$, the i -th input and output of Π are denoted by $(I_0(i), \dots, I_{k-1}(i))$ and $(S_0(i), \dots, S_{k-1}(i))$ respectively. Since the analysis is on the inverse of Π , in the remaining of the chapter, “input” refers to $(S_0(i), \dots, S_{k-1}(i))$ while “output” refers to $(I_0(i), \dots, I_{k-1}(i))$. In this chapter, five types of Generalized Feistel schemes are considered.

- Type-1 Feistel Schemes
- Type-2 Feistel Schemes
- Type-3 Feistel Schemes
- Alternating Feistel Schemes
- Unbalanced Feistel Schemes

In this chapter, differential analysis on the inverse of Π is considered. So the attack starts with the image $(S_0(i), \dots, S_{k-1}(i))$ and the differential path is built to the preimage, $(I_0(i), \dots, I_{k-1}(i))$.

3.2.2 Random Variable

Given a random variable X , denote by $E(X)$, $V(X)$, $\sigma(X)$ the expected value, variance and standard deviation of X respectively. Note that $V(X) = E(X^2) - E(X)^2$ and $\sigma(X) = \sqrt{V(x)}$.

Now given n random variables $X_1, \dots, X_n$, define the covariance of X_i and X_j as $Cov(X_i, X_j) = E(X_i X_j) - E(X_i)E(X_j)$. A simple calculation of the definition yields $V(\sum_{i=1}^n X_i) = \sum_{i=1}^n V(X_i) + \sum_{i \neq j, 1 \leq i, j \leq n} Cov(X_i, X_j)$.

Proposition 3.2.1. [105] Let X and Y be two random variables. X is said to be distinguishable from Y if $|E(X) - E(Y)| \geq \max(\sigma(X), \sigma(Y))$.

More specifically, let E_X, E_Y be the expected values of X and Y respectively while σ_X, σ_Y being the standard deviations of X and Y respectively. Without loss of generality, let $E_X < E_Y$. Then, if $E_Y - E_X \geq \max(\sigma(X), \sigma(Y))$,

$$1. \Pr\left(X \geq \frac{E_X + E_Y}{2}\right) \leq 0.30854$$

$$2. \Pr\left(Y \leq \frac{E_X + E_Y}{2}\right) \leq 0.30854.$$

Proof. We only prove the first claim since the second one can be proved by using the same method. A simple calculation tells us that:

$$\begin{aligned} \Pr\left(X \geq \frac{E_X + E_Y}{2}\right) &= \Pr\left(X - E_X \geq \frac{E_Y - E_X}{2}\right) \\ &\leq \Pr\left(X - E_X \geq \frac{\sigma_X}{2}\right) \\ &= \Pr\left(\frac{X - E_X}{\sigma_X} \geq \frac{1}{2}\right) \end{aligned}$$

Now by Central Limit Theorem, $\frac{X - E_X}{\sigma_X}$ can be approximated by a standard normal distribution. Hence by using this approximation and the standard normal distribution table, we get the upper bound claimed. $\square$

3.3 Overview of [105]

In this section, we provide a summary of the result provided in [105]. In the analysis done in the remaining part of this chapter has been done and compared to the result in [105]. The following table has been extracted from [105] to summarize the number of rounds distinguishable from random permutation in terms of the number of branches, k .

The main idea of the attack used in [105] is similar to what we considered in this chapter. However, instead of considering the backward direction for each type, they are considering the forward direction analysis.

The following table has been extracted from [105] to summarize the number of rounds distinguishable from random permutation in terms of the number of branches, k .

Table 3.1: Table of Complexity Distinguishing Attack on Various Attack Scenario and Generalized Feistel Schemes

Structure	KPA	CPA
Type-1	$2k^2 + 2k - 2$	$2k^2 + k - 1$
Type-2	$2k + 2$	$2k + 1$
Type-3	$k + \lceil \frac{k}{2} \rceil + 1$	$k + 1$
Alternating	$3k$	$3k$

3.4 Attack Overview

In this chapter, we exploit the asymmetry of the scheme by considering the backward differential diffusion. The clearest evidence of this can be found in Type-1 Feistel Scheme. The form of the scheme allows the diffusion to be slower in backward direction. More specifically, when a difference is introduced in one of the branches through F_i , instead of having the difference being input to another F_i directly in the next rounds, it will pass through $k - 1$ rounds without going through any F_i .

We will discuss two types of improvements:

- Unconstrained environment; We aim for a better lower bound for the maximum number of rounds distinguishable than the bound given in [105]. By unconstrained environment, we mean that analysis considered in this environment aims to distinguish more rounds than the previous results with complexity strictly less than $2^k n$ where k is the number of sub-blocks and n is the number of bit in each sub-blocks.

- Constrained environment; There are two possible forms of this improvement. Firstly, we aim to improve number of rounds that can be distinguished in the backward direction given the same complexity as the distinguisher given in [105]. Secondly, given the same number of rounds, we aim to reduce the complexity to distinguish the GFS from a random permutation.

Our analysis uses m plaintext-ciphertext pairs and considers the expected number of pairs $\mathcal{N}$ that satisfies certain conditions depending on the scheme analysed. Let $\mathcal{N}_{\text{perm}}$ be the value of $\mathcal{N}$ for a random permutation and $\mathcal{N}_{\mathcal{F}}$ be the value $\mathcal{N}$ for $\mathcal{F}$, the r -round Generalized Feistel Scheme. We use this information to calculate the maximum number of rounds r such that $\mathcal{N}_{\text{perm}}$ is distinguishable from $\mathcal{N}_{\mathcal{F}}$.

The function F_i (or $F_{(i,j)}$) used in the round function of GFS is assumed to be ideal keyed functions. Given the input, the output is a random n -bit string. Similarly, since it is ideal, given a nonzero input difference, the output difference is uniformly distributed.

Furthermore, let $\mathfrak{J}_1$ and $\mathfrak{J}_2$ be two distinct indices of the round function in the same cryptosystem (I_j can be a single integer or a pair or integers depending on the GFN we are considering). Given two different indices values $\mathfrak{J}_1$ and $\mathfrak{J}_2$, we also assume that $F_{\mathfrak{J}_1}$ and $F_{\mathfrak{J}_2}$ are independent from each other. Hence given the same input (or output) difference ΔS of $F_{\mathfrak{J}_1}$ and $F_{\mathfrak{J}_2}$, we can further assume that $F_{\mathfrak{J}_2}(\Delta S)$ is uniformly distributed even assuming that $F_{\mathfrak{J}_1}(\Delta S)$ is already known.

Intuitively, what we are trying to do is to first express the the difference in the preimage as functions of the difference in the image and the difference of some intermediate state. We express the preimage state as functions of the image strings since we want to launch a backward differential trail instead of the forward trail. To do this, we partition the strings to k sub-blocks and we express the equations as functions of the substrings.

Having these expressions, we can then choose carefully the input and output difference (truncated differences) to maximize the probability for the specified input

difference focusing in some of the sub-blocks to lead to the output difference chosen. To calculate the success probability, we consider the number of ciphertext pairs with specified difference that can lead to the plaintext pairs with the chosen input difference in two scenarios; when the function is a random permutation, denoted by $\mathcal{N}_{\text{perm}}$ and when the function is in the form of the Generalized Feistel Network considered, denoted by $\mathcal{N}_{\mathcal{F}}$.

Now having the expected values and variances of both $\mathcal{N}_{\text{perm}}$ and $\mathcal{N}_{\mathcal{F}}$, those four values will be functions of the round number r and number of ciphertext pairs with the chosen difference m . By Definition 3.2.1, $\mathcal{N}_{\mathcal{F}}$ is distinguishable from $\mathcal{N}_{\text{perm}}$ if $|E(\mathcal{N}_{\mathcal{F}}) - \mathcal{N}_{\text{perm}}| \geq \max(\sigma(\mathcal{N}_{\mathcal{F}}), \sigma(\mathcal{N}_{\text{perm}}))$. So using this inequality, we obtain a relation between the number of rounds r and the number of ciphertext pairs m . This will give us a lower bound of m given r . Since we want the distinguisher to be useful, we require m to be less than the total number of possible ciphertext pairs. In the case of known ciphertext attack, this means that we need $m \leq 2^{kn}$. This gives us an upper bound of round number r such that $\mathcal{F}$ is distinguishable from a random permutation using this backward differential attack.

In the following sections of this chapter, we perform this attack on different types of Generalized Feistel Networks discussed in Subsection 3.2.1.

3.5 Type-1 Feistel Scheme

Figure 3.1: Round i of Type-1 Feistel Scheme

3.5.1 Analysis of the Type-1 Feistel Scheme

For this analysis, we assume the number of rounds is $r = ak + b$ where k is the number of branches in the scheme and a and b are non-negative integers satisfying $0 \leq b \leq k - 1$ and $k \geq 3$. We will be using the notation described in the previous section for our analysis, namely $I_0, \dots, I_{k-1}$ for the k sub-blocks pre-image of Π while $S_0, \dots, S_{k-1}$ is used to denote the k sub-blocks image of Π . In this section we discuss in detail how we build the relations between the sub-blocks, then we discuss how we choose the differential trail. Having the differential trail, the expected value and the variance of the trail when Π is random permutation and a type-1 Generalized Feistel Scheme are calculated. This in turns tells us the maximum number of rounds that is distinguishable from the random permutation using the chosen differential.

Let X_i be the intermediate variables obtained in the second branch (indexed as 1) after the i -th round in the backward direction. By definition of the round function of Type-1 Feistel Scheme, we have the following relations:

- $X_0 = S_1$,
- $X_1 = S_0 \oplus F_{ak+b}(S_{k-1})$,
- For $t = 2, \dots, k - 1$, $X_t = S_{k+1-t} \oplus F_{ak+b-(t-1)}(S_{k-t})$,
- For $t \geq k$, $X_t = X_{t-k} \oplus F_{ak+b-(t-1)}(X_{t-(k-1)})$, Note that F_r is always used with input $X_{ak+b-r-k+2}$.
- For $r \geq k - 1$, the input of the r -th round in the backward direction is $(X_{r-(k-1)}, X_r, X_{r-1}, \dots, X_{r-(k-2)})$.

After $r = ak + b$ rounds, the state becomes $(I_0, \dots, I_{k-1})$ where $I_0 = X_{(ak+b)-(k-1)}$ and for $i = 1, \dots, k - 1$, $I_i = X_{(ak+b)-(i-1)}$. The following equalities can then be derived using the relations established above:

- $I_0 = X_{b+1} \oplus \bigoplus_{i=0}^{a-2} F_{(a-i-1)k}(X_{ik+b+2})$,

- For $j \in \{1, \dots, \min(k-1, b+1)\}$,

$$I_j = X_{b+1-j} \oplus \bigoplus_{i=0}^{a-1} F_{(a-i-1)k+j}(X_{ik+(b+2-j)}),$$

- For $j \in \{\min(k-1, b+1) + 1, \dots, k-1\}$,

$$I_j = X_{k+b+1-j} \oplus \bigoplus_{i=0}^{a-2} F_{(a-i-2)k+j}(X_{ik+(k+b+2-j)}).$$

In particular, for I_1 ,

$$\begin{aligned} I_1 &= X_b \oplus \bigoplus_{i=0}^{a-1} F_{(a-i-1)k+1}(X_{ik+(b+1)}) \\ &= X_b \oplus \bigoplus_{i=0}^{a-2} F_{(a-i-1)k+1}(X_{ik+(b+1)}) \oplus F_1(I_0) \\ &= \begin{cases} S_1 \oplus \bigoplus_{i=0}^{a-2} F_{(a-i-1)k+1}(X_{ik+1}) \oplus F_1(I_0) & \text{if } b = 0 \\ S_0 \oplus F_{ak+1}(S_{k-1}) \oplus \bigoplus_{i=0}^{a-2} F_{(a-i-1)k+1}(X_{ik+2}) \oplus F_1(I_0) & \text{if } b = 1 \\ S_{k+1-b} \oplus F_{ak+1}(S_{k-b}) \oplus \bigoplus_{i=0}^{a-2} F_{(a-i-1)k+1}(X_{ik+(b+1)}) \\ \quad \oplus F_1(I_0) & \text{otherwise.} \end{cases} \end{aligned}$$

We can further expand the sum by noting that when $i = 0$, the summand is $F_{(a-i-1)k+1}(X_{b+1})$ and

$$X_{b+1} = \begin{cases} S_0 \oplus F_{ak}(S_{k-1}) & \text{if } b = 0 \\ S_{k-b} \oplus F_{ak}(S_{k-b-1}) & \text{if } 1 \leq b \leq k-2 \\ S_1 \oplus F_{ak}(S_0 \oplus F_{ak+(k-1)}(S_{k-1})) & \text{if } b = k-1. \end{cases}$$

So in any value of $b \in \{0, \dots, k-1\}$, we can express I_1 as a function of several sub-blocks of the output $S_j, F_1(I_0)$ and $a-2$ terms determined by intermediate variables. More specifically, for $b \in \{0, \dots, k-1\}$, we have:

1. When $b = 0$,

$$\begin{aligned} I_1 \oplus S_1 \oplus F_1(I_0) &= \\ \bigoplus_{i=1}^{a-2} F_{(a-i-1)k+1}(X_{ik+1}) \oplus F_{(a-1)k+1}(S_0 \oplus F_{ak}(S_{k-1})), \end{aligned} \quad (3.5.1)$$

2. When $b = 1$,

$$I_1 \oplus S_0 \oplus F_1(I_0) = \bigoplus_{i=1}^{a-2} F_{(a-i-1)k+1}(X_{ik+2}) \oplus F_{(a-1)k+1}(S_{k-1} \oplus F_{ak}(S_{k-2})), \quad (3.5.2)$$

3. When $2 \leq b \leq k-2$,

$$I_1 \oplus S_{k+1-b} \oplus F_1(I_0) = \bigoplus_{i=1}^{a-2} F_{(a-i-1)k+1}(X_{ik+2}) \oplus F_{(a-1)k+1}(S_{k-b} \oplus F_{ak}(S_{k-b-1})), \quad (3.5.3)$$

4. When $b = k-1$,

$$I_1 \oplus S_2 \oplus F_1(I_0) = \bigoplus_{i=1}^{a-2} F_{(a-i-1)k+1}(X_{ik+1}) \oplus F_{(a-1)k+1}(S_1 \oplus F_{ak}(S_0 \oplus F_{(a+1)k-1}(S_{k-1}))). \quad (3.5.4)$$

To choose the truncated differential for each case, we try to utilize Equations 3.5.1, 3.5.2, 3.5.3 and 3.5.4. The main idea is we set the left hand side of the equation corresponding to any two plaintext-ciphertext pairs that we choose to agree. This gives us the relation we need for the plaintext-ciphertext pairs.

In particular, out of the m plaintext-ciphertext pairs, we count the number of $(s, t), 1 \leq s < t \leq m$ such that

$$\begin{aligned} 1. \quad I_0(s) &= I_0(t) \\ 2. \quad I_1(s) \oplus I_1(t) &= \begin{cases} S_1(s) \oplus S_1(t) & \text{if } b = 0 \\ S_0(s) \oplus S_0(t) & \text{if } b = 1 \\ S_{k+1-b}(s) \oplus S_{k+1-b}(t) & \text{if } 2 \leq b \leq k-1. \end{cases} \end{aligned} \quad (3.5.5)$$

Note that in any of the equation that we have, we still have one term containing some sub-blocks of the ciphertext. To increase the probability that the equation is satisfied, we can set it to have no difference in any of the plaintext-ciphertext pairs. So in particular, in the CCA model, pick m different ciphertext such that:

- If $b = 0$, pick all the ciphertext with fixed values of $S_0(s)$ and $S_{k-1}(s)$. Hence in the CCA attack, $m \leq 2^{(k-2)n}$.
- If $b = 1, \dots, k-2$, fix the values of $S_{k-b}(s)$ and $S_{k-b-1}(s)$ for all $s = 0, \dots, m-1$. Again, in the CCA attack, $m \leq 2^{(k-2)n}$.
- If $b = k-1$ and $k \geq 4$, fix the values of $S_0(s), S_1(s)$ and $S_{k-1}(s)$ for $s = 0, \dots, m-1$. In this case, the CCA attack must have m to be at most $2^{(k-3)n}$.

So in summary, the differential trail for $r = ak+b$ rounds where $b \in \{0, \dots, k-1\}$ is as follows:

1. In KPA setting, the input (ciphertext) differential is $(\nabla_0, \dots, \nabla_{k-1})$ while the output (plaintext) differential is $(\Delta_0, \dots, \Delta_{k-1})$ where it satisfies the following equations:

- $\Delta_0 = 0$.
- $\Delta_1 = \nabla_{(k+1-b) \pmod k}$
- All other sub-blocks difference is arbitrary, which we denote by $\star$.

Let $\mathcal{N}_{F,M}$ be the random variable representing the number of sets of two plaintext-ciphertext pairs that satisfy the conditions given by (3.5.5) for F representing the function used, which has value in the set $\{\text{perm}, \mathcal{F}\}$, and $M \in \{\text{KPA}, \text{CCA}\}$. $F = \text{perm}$ is used for the random permutation while $F = \mathcal{F}$ is used for the r -round Type-1 Feistel Scheme.

Now it is easy to see that the probability that the requirement set above to be true is equal to the probability that the right hand side of the equations to agree, which can be computed since we can assume all the X_i is uniformly and independently distributed by the ideality of the round function (which has been discussed in Section 3.4).

Calculating the expected values and variance of the random variables,

$$E(\mathcal{N}_{(\text{perm}, \text{KPA})}), E(\mathcal{N}_{(\text{perm}, \text{CCA})}), V(\mathcal{N}_{(\text{perm}, \text{KPA})}), V(\mathcal{N}_{(\text{perm}, \text{CCA})})$$

are all approximately $\frac{m^2}{2 \cdot 2^{2n}}$. Calculating the random variables corresponding to $\mathcal{F}$, the expected values and variances are summarised in Table 3.2. The details on the calculation of the expected values and variances of $\mathcal{N}_{\mathcal{F}, \text{CCA}}$ for $b = 0$ can be found in Subsection 3.5.2. The other results can be calculated using the same method.

Table 3.2: Expected Value and Variance of random variables concerning Type-1 Feistel Schemes

b	Attack Model	Expected Value	Variance	Maximum value of m
0	KPA	$\frac{m^2}{2} \left(\frac{1}{2^{2n}} + \frac{1}{2^{an}} \right)$	$\frac{m^2}{2 \cdot 2^{2n}}$	2^{kn}
	CCA	$\frac{m^2}{2} \left(\frac{1}{2^{2n}} + \frac{1}{2^{(a-1)n}} \right)$	$\frac{m^2}{2 \cdot 2^{2n}}$	$2^{(k-2)n}$
$1 \leq b \leq k-2$	KPA	$\frac{m^2}{2} \left(\frac{1}{2^{2n}} + \frac{1}{2^{(a+1)n}} \right)$	$\frac{m^2}{2 \cdot 2^{2n}}$	2^{kn}
	CCA	$\frac{m^2}{2} \left(\frac{1}{2^{2n}} + \frac{1}{2^{(a-1)n}} \right)$	$\frac{m^2}{2 \cdot 2^{2n}}$	$2^{(k-2)n}$
$k-1$	KPA	$\frac{m^2}{2} \left(\frac{1}{2^{2n}} + \frac{1}{2^{(a+1)n}} \right)$	$\frac{m^2}{2 \cdot 2^{2n}}$	2^{kn}
	CCA	$\frac{m^2}{2} \left(\frac{1}{2^{2n}} + \frac{1}{2^{(a-1)n}} \right)$	$\frac{m^2}{2 \cdot 2^{2n}}$	$2^{(k-3)n}$

In the KPA model, the maximum number of rounds is k^2 where from $k(k-1)+1$ up to k^2 rounds, the complexity is $\sqrt{2} \cdot 2^{(k-1)n}$. Furthermore, in the CCA model, the maximum number of rounds distinguishable is $k(k-1) + k - 2 = k^2 - 2$ rounds with complexity $\sqrt{2} \cdot 2^{(k-3)n}$.

3.5.2 Calculation of Expected Value and Variance of $\mathcal{N}_{(\mathcal{F}, \text{CCA})}$ where $\mathcal{F}$ is an ak -round Type-1 Feistel Scheme

In this subsection we give an example of how to compute the expected value and variance of random variables considered in the chapter. Although this is just for an ak round Type-1 Feistel scheme, the same argument can be used to compute all other cases. Note that we are not considering the calculation of $\mathcal{N}_{(\text{perm}, M)}$ for any $M \in \{\text{KPA}, \text{CCA}\}$ since in a random permutation function perspective, the input and output are interchangeable. Hence the calculation will exactly be the same as the

one given in [105].

First recall the problem that we are trying to solve. Suppose there are m plaintext-ciphertext pairs $(I_0(s), \dots, I_{k-1}(s)), (S_0(s), \dots, S_{k-1}(s)), s \in 1, \dots, m$ such that for all $s \neq t, S_0(s) = S_0(t), S_{k-1}(s) = S_{k-1}(t)$. Given this, $\mathcal{N}_{(\mathcal{F}, \text{CCA})}$ is defined to be the random variable storing the size of the set $\{(i, j) : 1 \leq i < j \leq m, I_0(i) = I_0(j), I_1(i) \oplus I_1(j) = S_1(i) \oplus S_1(j)\}$ where the plaintext-ciphertext pairs are generated through function $\mathcal{F}$ under the restriction mentioned above. Now

$$\begin{aligned} \mathcal{N}_{(\mathcal{F}, \text{CCA})} &= |\{(i, j) : 1 \leq i < j \leq m, I_0(i) = I_0(j), \\ &\quad I_1(i) \oplus I_1(j) = S_1(i) \oplus S_1(j)\}| \\ &= \sum_{i < j} f(i, j) \end{aligned}$$

where $f(i, j)$ is defined as:

$$f(i, j) = \begin{cases} 1, & \text{if } I_0(i) = I_0(j), I_1(i) \oplus I_1(j) = S_1(i) \oplus S_1(j) \\ 0, & \text{otherwise} \end{cases}.$$

So by linearity of expected value and definition of variance:

•

$$\begin{aligned} E(\mathcal{N}_{(\mathcal{F}, M)}) &= \sum_{i < j} E(f(i, j)) \\ &= \sum_{i < j} 1 \cdot P(f(i, j) = 1) + 0 \cdot P(f(i, j) = 0) \\ &= \sum_{i < j} P(I_0(i) = I_0(j), I_1(i) \oplus I_1(j) = S_1(i) \oplus S_1(j)) \end{aligned}$$

•

$$\begin{aligned}
V(\mathcal{N}_{(F,M)}) &= \sum_{i < j} V(f(i, j)) + \sum_{(i,j) \neq (r,s): i < j, r < s, 1 \leq i, j, r, s \leq m} \text{Cov}(f(i, j), f(r, s)) \\
&= \sum_{i < j} [E(f(i, j)^2) - E(f(i, j))^2] \\
&\quad + \sum_{(i,j) \neq (r,s): i < j, r < s, 1 \leq i, j, r, s \leq m} [E(f(i, j)f(r, s)) - E(f(i, j))E(f(r, s))] \\
&= \sum_{i < j} [P(f(i, j) = 1) - P(f(i, j) = 1)^2] \\
&\quad + \sum_{(i,j) \neq (r,s): i < j, r < s, 1 \leq i, j, r, s \leq m} [P(f(i, j) = 1, f(r, s) = 1) \\
&\quad - P(f(i, j) = 1)P(f(r, s) = 1)]
\end{aligned}$$

Now recall that we have some relations between S and I from $\mathcal{F}$ that can be found in Section 3.5. Using this relation and noting the values fixed, the condition for $f(i, j)$ to be 1 can be rewritten as

- $I_0(i) = I_0(j)$
- $\sum_{t=1}^{a-2} F_{(a-t-1)k+1}(X_{tk+1}(i)) = \sum_{t=1}^{a-2} F_{(a-t-1)k+1}(X_{tk+1}(j)).$

For simplicity of notation, define two variables for any i as follows:

1. $Y(i) := (X_{tk+1}(i))_{t=1}^{a-2}$ and
2. $Z(i) := \sum_{t=1}^{a-2} F_{(a-t-1)k+1}(X_{tk+1}(i)).$

First we calculate $E(\mathcal{N}_{(\mathcal{F}, \text{CCA})})$. As noted above, to find this, the value of $P(f(i, j) = 1)$ needs to be computed. Fix i, j such that $1 \leq i < j \leq m$. Then by the ideal as-

sumption of F_i and law of total probability:

$$\begin{aligned}
P(I_0(i) = I_0(j), Z(i) = Z(j)) &= P(I_0(i) = I_0(j)) \cdot P(Z(i) = Z(j)) \\
&= P(I_0(i) = I_0(j)) \cdot [P(Z(i) = Z(j)|Y(i) = Y(j)) \\
&\quad \cdot P(Y(i) = Y(j)) + P(Z(i) = Z(j)|Y(i) \neq Y(j)) \\
&\quad \cdot P(Y(i) \neq Y(j))] \\
&= \frac{1}{2^n} \cdot \left[1 \cdot \frac{1}{2^{(a-2)n}} + \frac{1}{2^n} \cdot \left(1 - \frac{1}{2^{(a-2)n}} \right) \right] \\
&= \left(\frac{1}{2^{2n}} + \frac{1}{2^{(a-1)n}} - \frac{1}{2^{an}} \right)
\end{aligned}$$

This gives us three consequences:

1. $E(\mathcal{N}_{(\mathcal{F}, \text{CCA})}) = \frac{m(m-1)}{2} \left(\frac{1}{2^{2n}} + \frac{1}{2^{(a-1)n}} - \frac{1}{2^{an}} \right) \approx \frac{m^2}{2} \left(\frac{1}{2^{2n}} + \frac{1}{2^{(a-1)n}} \right)$
2. $V(f(i, j)) = \frac{1}{2^{2n}} + \frac{1}{2^{(a-1)n}} - \left(\frac{1}{2^{2n}} + \frac{1}{2^{(a-1)n}} \right)^2$ which can be simplified to
$$\begin{aligned}
V(f(i, j)) &= \frac{1}{2^{2n}} - \frac{1}{2^{4n}} + \frac{1}{2^{(a-1)n}} - \frac{1}{2^{an}} - \frac{2}{2^{(a+1)n}} + \frac{2}{2^{(a+2)n}} - \frac{1}{2^{(2a-2)n}} \\
&\quad + \frac{2}{2^{(2a-1)n}} - \frac{1}{2^{2an}}.
\end{aligned}$$
3. For any i, j, r, s such that $(i, j) \neq (r, s), i < j, r < s$,

$$\begin{aligned}
E(f(i, j))E(f(r, s)) &= E(f(i, j))^2 = \frac{1}{2^{4n}} + \frac{2}{2^{(a+1)n}} - \frac{2}{2^{(a+2)n}} + \frac{1}{2^{(2a-2)n}} \\
&\quad - \frac{2}{2^{(2a-1)n}} + \frac{1}{2^{2an}}.
\end{aligned}$$

Lastly the variance of $\mathcal{N}_{(\mathcal{F}, \text{CCA})}$ is computed. By the consequences mentioned above, the variance can be computed once the values of $E(f(i, j) \cdot f(r, s)) = P(f(i, j) = 1, f(r, s) = 1)$ for $(i, j) \neq (r, s)$ is found. For this there are two cases to consider depending on the size of $\{i, j, r, s\}$ since it can be either 3 or 4.

3.5.2.1 $|\{i, j, r, s\}| = 4$

Recall from before that $f(i, j) = 1$ and $f(r, s) = 1$ if and only if $I_0(i) = I_0(j), I_0(r) = I_0(s), Z(i) = Z(j), Z(r) = Z(s)$. Divide the calculation into 6 different cases using law of total probability.

- Case 1: $I_0(i) = I_0(j), I_0(r) = I_0(s), Y(i) = Y(j), Y(r) = Y(s)$. In this case, the probability is $\left(\frac{1}{2^n}\right)^{2(a-1)} = \frac{1}{2^{2(a-1)n}}$.
- Case 2: $I_0(i) = I_0(j), I_0(r) = I_0(s), Y(i) = Y(j), Y(r) \neq Y(s), Z(r) = Z(s)$. In this case, the probability is

$$\frac{1}{2^{2n}} \cdot \frac{1}{2^{(a-2)n}} \cdot \left(1 - \frac{1}{2^{(a-2)n}}\right) \cdot \frac{1}{2^n} = \frac{1}{2^{(a+1)n}} - \frac{1}{2^{(2a-1)n}}.$$

- Case 3: $I_0(i) = I_0(j), I_0(r) = I_0(s), Y(i) \neq Y(j), Z(i) = Z(j), Y(r) = Y(s)$. This case is equivalent to case 2. Hence the probability is

$$\frac{1}{2^{(a+1)n}} - \frac{1}{2^{(2a-1)n}}.$$

- Case 4: $I_0(i) = I_0(j), I_0(r) = I_0(s), Y(i) \neq Y(j), Z(i) = Z(j), Y(r) = Y(i), Y(s) = Y(j)$. The probability of this case is

$$\frac{1}{2^{2n}} \cdot \left(1 - \frac{1}{2^{(a-2)n}}\right) \cdot \frac{1}{2^{2(a-2)n}} \cdot \frac{1}{2^n} = \frac{1}{2^{(2a-1)n}} - \frac{1}{2^{(3a-3)n}}.$$

- Case 5: $I_0(i) = I_0(j), I_0(r) = I_0(s), Y(i) \neq Y(j), Z(i) = Z(j), Y(r) = Y(j), Y(s) = Y(i)$. This case is equivalent to case 4. Hence the probability is

$$\frac{1}{2^{(2a-1)n}} - \frac{1}{2^{(3a-3)n}}$$

- Case 6: $I_0(i) = I_0(j), I_0(r) = I_0(s), Y(i) \neq Y(j), Z(i) = Z(j), Y(r) \neq Y(s)$ and it is not in case 4 or 5. The probability here can be calculated as

$$\frac{1}{2^{2n}} \cdot \left[\left(1 - \frac{1}{2^{(a-2)n}}\right)^2 - 2 \cdot \left(1 - \frac{1}{2^{(a-2)n}}\right) \cdot \left(\frac{1}{2^{(a-2)n}}\right)^2 \right] \cdot \frac{1}{2^{2n}}$$

which can be simplified to

$$\frac{1}{2^{4n}} - \frac{2}{2^{(a+2)n}} - \frac{1}{2^{2an}} + \frac{2}{2^{(3a-2)n}}.$$

So adding all this up gives us $E(f(i, j) \cdot f(r, s))$. Hence when $|\{i, j, r, s\}| = 4$,

$$Cov(f(i, j), f(r, s)) = \frac{2}{2^{(2a-1)n}} - \frac{2}{2^{2an}} + \frac{2}{2^{(3a-2)n}} - \frac{2}{2^{(3a-3)n}}.$$

Now we count the number of $(i, j), (r, s)$ in this case. Note that given $\{a_1, a_2, a_3, a_4\}$ of cardinality 4, it is easy to see that there are 6 ways to assign the 4 values to i, j, r, s satisfying $(i, j) \neq (r, s), i < j, r < s$. Hence we have $6\binom{m}{4}$ such $(i, j), (r, s)$.

3.5.2.2 $|\{i, j, r, s\}| = 3$

Without loss of generality, assume $i < j = r < s$. All the other cases can be handled in a similar manner.

- Case 1: $I_0(i) = I_0(j) = I_0(s), Y(i) = Y(j) = Y(s)$. The probability is

$$\frac{1}{2^{(2a-2)n}}.$$

- Case 2: $I_0(i) = I_0(j) = I_0(s), |\{Y(i), Y(j), Y(s)\}| = 2, Z(i) = Z(j) = Z(s)$ which can happen in 3 different ways. In any of these ways, the probability is

$$\frac{1}{2^{2n}} \cdot \frac{1}{2^{(a-2)n}} \cdot \left(1 - \frac{1}{2^{(a-2)n}}\right) \cdot \frac{1}{2^n} = \frac{1}{2^{(a+1)n}} - \frac{1}{2^{(2a-1)n}}.$$

- Case 3: $I_0(i) = I_0(j) = I_0(s), |\{Y(i), Y(j), Y(s)\}| = 3, Z(i) = Z(j) = Z(s)$. In this case, the probability is

$$\frac{1}{2^{2n}} \cdot \left(1 - \frac{1}{2^{(a-2)n}}\right) \cdot \left(1 - \frac{2}{2^{(a-2)n}}\right) \cdot \frac{1}{2^{2n}} = \frac{1}{2^{4n}} - \frac{3}{2^{(a+2)n}} + \frac{2}{2^{2an}}.$$

So adding all the cases and then subtracting $E(f(i, j))^2$ from it gives us:

$$Cov(f(i, j), f(j, r)) = \frac{1}{2^{(a+1)n}} - \frac{1}{2^{(a+2)n}} - \frac{1}{2^{(2a-1)n}} + \frac{1}{2^{2an}}.$$

Lastly, we find the number of such $(i, j), (r, s)$ such that $(i, j) \neq (r, s), i < j, r < s, |\{i, j, r, s\}| = 3$. Given $\{a_1, a_2, a_3\}$, it is easy to see that there are 6 different assignments of these three values to i, j, r, s that satisfies the above requirements. Hence the multiplier of this case is $6\binom{m}{3}$.

So combining all the values obtained above, $V(\mathcal{N}_{(\mathcal{F}, \text{CCA})})$ can be computed. A simple calculation and simplification of the terms gives the value that we claimed in Section 3.5,

$$V(\mathcal{N}_{(\mathcal{F}, \text{CCA})}) \approx \frac{m^2}{2 \cdot 2^{2n}}.$$

Using the proposition of distinguishability of two random variables given in the preliminaries, the result is provided in Table 3.3.

Table 3.3: Summary of Distinguishability of Type-1 Feistel Scheme

b	Model	Complexity of distinguishing $ak + b$ rounds	Maximum a
0	KPA	$\sqrt{2} \cdot 2^{(a-1)n}$	k
	CCA	$\sqrt{2} \cdot 2^{(a-2)n}$	$k - 1$
$1 \leq b \leq k - 2$	KPA	$\sqrt{2} \cdot 2^{an}$	$k - 1$
	CCA	$\sqrt{2} \cdot 2^{(a-2)n}$	$k - 1$
$k - 1$	KPA	$\sqrt{2} \cdot 2^{an}$	$k - 1$
	CCA	$\sqrt{2} \cdot 2^{(a-2)n}$	$k - 2$

3.5.3 Comparison with Existing Result from [105]

To compare with the result given in [105] first note that there are some constants multiplier that are omitted in [105]. More specifically, all the expected values and variances should be multiplied by $\frac{1}{2}$. This constant adjustment comes from the fact that given m plaintext-ciphertext pairs, the number of sets of 2 distinct pairs should be $\frac{m(m-1)}{2} \approx \frac{m^2}{2}$ instead of m^2 . Although the constant multiplier is very close to one compared to 2^n , it affects the maximum number of rounds that can be distinguished in the KPA and CPA model. This is because all the complexities of distinguishers should be multiplied by a factor of $\sqrt{2}$. The existence of this factor makes it impossible for a to reach the maximum number given in [105]. For $ak - 2$ rounds distinguished in KPA model, the complexity should be $\sqrt{2} \cdot 2^{(a-2)n}$. Hence the maximum number

of rounds that can be distinguished in KPA model is $k^2 + k - 2$ rounds instead of $k^2 + 2k - 2$ rounds. Similarly, for $ak - 1$ rounds to be distinguishable in CPA, the complexity is again $\sqrt{2} \cdot 2^{(a-2)n}$. This leads to the maximum number of rounds that is distinguishable in CPA model is $k^2 - 1$ rounds instead of $k^2 + k - 1$.

Note that in both cases, the maximum number of rounds distinguishable without any complexity constraint is still better in the forward direction. So in this section, the advantage of using the backward direction analysis in a constrained environment is discussed.

We compare the results in the CCA model presented above with the CPA model. When the complexity is $\sqrt{2} \cdot 2^{tn}$, in CPA model, the maximum number of rounds that is distinguishable is $(t + 2)k - 1$ rounds while in CCA model, the maximum number of rounds that is distinguishable is $(t + 2)k + (k - 2) = (t + 3)k - 2$ rounds which is an increase of $k - 1$ rounds. Suppose that we want to distinguish r rounds for some positive integer r . Table 3 of [105] (after the adjustment by a factor of $\sqrt{2}$) tells us that when $pk - (p - 2) = (p - 1)k + k - p + 2 \leq r \leq (p + 1)k - p = pk + k - p$, the complexity is $\sqrt{2} \cdot 2^{(p-2)n}$. Using the same bound for r , the complexity is $\sqrt{2} \cdot 2^{(p-3)n}$ when $r \leq pk - 1$ and $\sqrt{2} \cdot 2^{(p-2)n}$ when $r \geq pk$ (see Table 3.3). So the complexity is reduced by a factor of $\frac{1}{2^n}$ when $(p - 1)k + k - p + 2 \leq r \leq pk$ for any value of p . Now for all the following sections, since the method that is being used is exactly the same, we will not discuss in detail on how to choose the differential, the expected values and the distinguishability. Instead, only the final results will be stated and compared. We note that since we are using Proposition 3.2.1, which is also used in the analysis in [105], has success probability at least 70 percents.

Figure 3.2: Round i of Type-3 Feistel Scheme

3.6 Type-3 Feistel Scheme

3.6.1 Analysis of the Type-3 Feistel Scheme

As before, we denote the input as $S_0, \dots, S_{k-1}$. Define intermediate variables X_i such that $(X_{tk}, \dots, X_{tk+k-1})$ is the state value after t rounds. Assuming the number of rounds is r , for $0 \leq s \leq k-1$, $X_s = S_s$ and $X_{rk+s} = I_s$. Given the input of round c , $1 \leq c \leq r$, by definition:

$$X_{ck} = X_{(c-1)k+k-1}$$

$$X_{ck+s} = X_{(c-1)k+(s-1)} \oplus F_{(r+1-c, s-1)}(X_{ck+s-1}), \quad \forall 1 \leq s \leq k-1.$$

Let $r = ak + b$ for $0 \leq b \leq k-1$. In this chapter, we only consider $a = 1$ and $b > 0$. Expanding the equation for $X_{(k+b)k+s}$ using the equation given above, the following can then be derived:

- When $b = s$,

$$X_{(k+b)k+s} = \bigoplus_{i=0}^{b-1} F_{(i+1, s-1-i)}(X_{(k+b-i)k+(s-1-i)})$$

$$\oplus \bigoplus_{i=0}^{k-2} F_{(i+b+2, k-2-i)}(X_{(k-1-i)k+(k-2-i)}) \oplus S_0.$$

- When $b = s + 1 \leq k - 1$,

$$X_{(k+b)k+s} = \bigoplus_{i=0}^{s-1} F_{(i+1,s-1-i)}(X_{(k+b-i)k+(s-1-i)}) \oplus F_{(b+1,k-2)}(X_{(k)k+k-2}) \\ \oplus \bigoplus_{i=0}^{k-3} F_{(i+b+2,k-3-i)}(X_{(k-1-i)k+k-3-i}) \oplus S_{k-1}.$$

- When $s + 1 < b \leq k - 1$,

$$X_{(k+b)k+s} = \bigoplus_{i=0}^{s-1} F_{(i+1,s-1-i)}(X_{(k+b-i)k+(s-1-i)}) \\ \oplus \bigoplus_{i=0}^{b-s-1} F_{(s+i+2,k-2-i)}(X_{(k+b-s-1-i)k+k-2-i}) \\ \oplus \bigoplus_{i=0}^{k-b+s-2} F_{(i+b+2,k-b+s-2-i)}(X_{(k-1-i)k+k-b+s-2-i}) \\ \oplus \bigoplus_{i=0}^{b-s-2} F_{(k+s+i+2,k-2-i)}(X_{(b-s-1-i)k+(k-2-i)}) \oplus S_{k-1}.$$

- When $s = b + 1 \leq k - 1$,

$$X_{(k+t)k+b} = \bigoplus_{i=0}^b F_{(i+1,b-i)}(X_{(k+b-i)k+b-i}) \\ \oplus \bigoplus_{i=0}^{k-3} F_{(i+3,k-2-i)}(X_{(k-2-i)k+(k-2-i)}) \oplus S_1.$$

- When $b + 1 < s \leq k - 1$,

$$X_{(k+b)k+s} = \bigoplus_{i=0}^b F_{(i+1,b-i)}(X_{(k+b-i)k+b-i}) \\ \oplus \bigoplus_{i=0}^{s-b-2} F_{(b+i+2,s-b-2-i)}(X_{(k-1-i)k+s-b-2-i}) \\ \oplus \bigoplus_{i=0}^{k-s+b-2} F_{(s+2+i,k-2-i)}(X_{(k-s+b-1-i)k+(k-2-i)}) \oplus S_{s-b}.$$

Let $b \in \{1, \dots, k - 1\}$. For the m plaintext-ciphertext pairs, the distinguishing attack counts the number of sets of two pairs $(j, j'), 1 \leq j < j' \leq m$ that satisfies the

following two conditions:

$$\begin{aligned} 1. \quad I_{(r-1)}(j) &= I_{(r-1)}(j') \\ 2. \quad I_r(j) \oplus I_r(j') &= S_0(j) \oplus S_0(j'). \end{aligned} \tag{3.6.1}$$

In the CCA model, fix the value of $S_{k-1}(j)$ of all the m ciphertexts. Hence $m \leq 2^{(k-1)n}$.

Calculating the random variables with the same method, $E(\mathcal{N}_{(\text{perm}, \text{KPA})})$, $V(\mathcal{N}_{(\text{perm}, \text{KPA})})$, $E(\mathcal{N}_{(\text{perm}, \text{CCA})})$, $V(\mathcal{N}_{(\text{perm}, \text{CCA})})$, $V(\mathcal{N}_{(\mathcal{F}, \text{KPA})})$ and $V(\mathcal{N}_{(\mathcal{F}, \text{CCA})})$ are all approximately $\frac{m^2}{2 \cdot 2^{2n}}$ while

$$E(\mathcal{N}_{(\mathcal{F}, \text{KPA})}) = \frac{m^2}{2} \left(\frac{1}{2^{2n}} + \frac{1}{2^{(k+r-2)n}} \right)$$

and

$$E(\mathcal{N}_{(\mathcal{F}, \text{CCA})}) = \frac{m^2}{2} \left(\frac{1}{2^{2n}} + \frac{1}{2^{(k+r-3)n}} \right).$$

In both KPA and CPA model, $\mathcal{F}$ is distinguishable from a random permutation when there are up to $k + 2$ rounds and the complexity to distinguish $k + b$ rounds are $\sqrt{2} \cdot 2^{(k+b-3)n}$ and $\sqrt{2} \cdot 2^{(k+b-4)n}$ respectively.

3.6.2 Comparison with Existing Result from [105]

Now we compare our result with the one given in [105]. First of all, note that in [105], there is a restriction that $\lfloor \frac{k}{2} \rfloor \geq 3$. This means that k needs to be at least 6 while the distinguisher proposed above can be used for all $k \geq 2$. So in any attack model, this analysis provides a new lower bound of the maximum number of distinguishable round for $2 \leq k \leq 5$. Furthermore, when $k \geq 6$, in KPA model, in both $k + 1$ and $k + 2$ rounds proposed above, the complexity is higher than the one given in [105]. The same thing happen in the distinguisher for $k + 1$ rounds in CCA model. However, the lower bound of maximum number of rounds distinguishable from random permutation in CCA model is increased to $k + 2$ from $k + 1$ proposed in [105].

3.7 Alternating Feistel Scheme

Figure 3.3: Round $2i - 1$ and Round $2i$ of Alternating Feistel Scheme

3.7.1 Analysis of Alternating Feistel Scheme

We divide this section to two cases based on the parity of the number of rounds. This is required due to the different round function in odd and even rounds.

3.7.1.1 Even Number of Rounds

Suppose that the number of rounds is $2r$. Let X_i be intermediate variables such that after $2t$ rounds the state value is

$$(X_{tk}, \dots, X_{tk+k-1}).$$

For any $0 \leq s \leq k-1$, $(I_s, S_s) = (X_{rk+s}, X_s)$. Then, given the state value after $2t$ rounds, $(X_{tk}, \dots, X_{tk+k-1})$ where $0 \leq t \leq r-1$, we have the following relations:

- $X_{(t+1)k} = X_{tk} \oplus F_{(r-t)}((X_{tk+s} \oplus F_{(r-t,s)}(X_{tk}))_{s=1}^{k-1})$ where

$$(Y_a)_{a=r}^s := (Y_r, Y_{r+1}, \dots, Y_s).$$

- $X_{(t+1)k+s} = X_{tk+s} \oplus F_{(r-t,s)}(X_{tk}), \forall s = 1, \dots, k-1.$

Then expand the equation for I_s :

- $I_0 = S_0 \oplus \bigoplus_{i=0}^{r-1} F_{(r-i)}((X_{ik+s} \oplus F_{(r-i,s)}(X_{ik}))_{s=1}^{k-1})$
- $\forall s \in \{1, \dots, k-1\},$

$$I_s = S_s \oplus \bigoplus_{i=0}^{r-1} F_{(r-i,s)}(X_{ik}) = S_s \oplus F_{(r,s)}(S_0) \oplus \bigoplus_{i=1}^{r-1} F_{(r-i,s)}(X_{ik}).$$

The distinguishing attack finds the number of sets of two plaintext ciphertext pairs $(p, q), 1 \leq p < q \leq m$ such that they satisfy the following conditions:

$$\begin{aligned} 1. \quad & I_0(p) = I_0(q) \\ 2. \quad & \forall s = 1, \dots, k-1, \quad I_b(p) \oplus I_b(q) = S_b(p) \oplus S_b(q). \end{aligned} \tag{3.7.1}$$

Furthermore, in CCA model, all the ciphertexts are chosen such that they have the same fixed value in $S_0(p)$. So we have, $m \leq 2^{(k-1)n}$.

Using the same calculation as before, $E(\mathcal{N}_{(\text{perm}, \text{KPA})}), V(\mathcal{N}_{(\text{perm}, \text{KPA})}), E(\mathcal{N}_{(\text{perm}, \text{CCA})}), V(\mathcal{N}_{(\text{perm}, \text{CCA})}), V(\mathcal{N}_{(\mathcal{F}, \text{KPA})})$ and $V(\mathcal{N}_{(\mathcal{F}, \text{KPA})})$ can all be approximated by $\frac{m^2}{2 \cdot 2^{kn}}$. Furthermore,

$$E(\mathcal{N}_{(\mathcal{F}, \text{KPA})}) = \frac{m^2}{2} \left(\frac{1}{2^{kn}} + \frac{1}{2^{rn}} \right) \text{ and } E(\mathcal{N}_{(\mathcal{F}, \text{CCA})}) = \frac{m^2}{2} \left(\frac{1}{2^{kn}} + \frac{1}{2^{(r-1)n}} \right).$$

Simplifying this, to distinguish $2r$ rounds, the complexity is $\sqrt{2} \cdot 2^{(r-\frac{k}{2})n}$ for KPA and $\sqrt{2} \cdot 2^{(r-\frac{k}{2}-1)n}$ for CCA. So when k is even, in both models, $\mathcal{F}$ can be distinguished from a random permutation when the round number is up to $3k-2$ with complexity $\sqrt{2} \cdot 2^{(k-1)n}$ and $\sqrt{2} \cdot 2^{(k-2)n}$ respectively. When k is odd, $\mathcal{F}$ can be distinguished from a random permutation when the round number is up to $3k-1$. In this case, the complexity is $\sqrt{2} \cdot 2^{(k-\frac{1}{2})n}$ and $\sqrt{2} \cdot 2^{(k-\frac{3}{2})n}$ respectively.

3.7.1.2 Odd Number of Rounds

Suppose that the number of rounds is $2r+1$ for some non-negative integers r . Let X_i be intermediate variables such that for any non-negative integer t , after $2t+1$ rounds,

the state value is $(X_{tk}, \dots, X_{tk+k-1})$. So $I_s = X_{rk+s}$ for all $0 \leq s \leq k-1$ while

$$X_s = \begin{cases} S_s, & \text{if } s = 1, \dots, k-1, \\ S_0 \oplus F_{r+1}(S_1, \dots, S_{k-1}) & \text{if } s = 0. \end{cases}$$

Following the expansion done before, the following equalities can be found:

- $I_0 = S_0 \oplus F_{r+1}(S_1, \dots, S_{k-1}) \oplus \bigoplus_{i=0}^{r-1} F_{(r-i)}((X_{ik+s} \oplus F_{(r-i,s)}(X_{ik}))_{s=1}^{k-1})$
- $\forall s = 1, \dots, k-1, I_s = S_s \oplus \bigoplus_{i=0}^{r-1} F_{(r-i,s)}(X_{ik}) = S_s \oplus F_{(r,s)}(S_0) \oplus \bigoplus_{i=1}^{r-1} F_{(r-i,s)}(X_{ik})$.

Because of this, all the distinguisher and calculation considered in the even number of rounds case can still be used in this case. Hence the complexity to distinguish $2r+1$ rounds is $\sqrt{2} \cdot 2^{(r-\frac{k}{2})n}$ for KPA and $\sqrt{2} \cdot 2^{(r-\frac{k}{2}-1)n}$ for CCA. When k is even, in both models, $\mathcal{F}$ can be distinguished from a random permutation when the round number is up to $3k-1$ with complexity $\sqrt{2} \cdot 2^{(k-1)n}$ and $\sqrt{2} \cdot 2^{(k-2)n}$ respectively. When k is odd, we can distinguish up to $3k$ rounds with complexity $\sqrt{2} \cdot 2^{(k-\frac{1}{2})n}$ and $\sqrt{2} \cdot 2^{(k-\frac{3}{2})n}$ respectively.

3.7.2 Comparison with Existing Result from [105]

We compare the result of previous subsection with the one given in Section 4.4 in [105]. As before, note all the expected values and variance should be multiplied by $\frac{1}{2}$, all the complexities should be multiplied by $\sqrt{2}$ and hence maximum number of rounds, in this case, should be decreased by 2. After this adjustment, to distinguish t rounds, the complexities are summarised in Table 3.4:

In both models, when the number of rounds is odd, the complexity is better than the forward direction, which is a reduction by a factor of $2^{\frac{3n}{2}}$. However, when the number of rounds is even, backward direction requires the same complexity in the KPA model. In the CCA model, the complexity of backward direction is reduced by a factor of 2^n .

Note that after the adjustment to the result in [105], backward differential analysis achieves 2 more rounds in both models, from $3k-2$ rounds to $3k$ rounds.

Table 3.4: Summary of Comparison of Alternating Feistel Schemes with fixed number of rounds t .

Parity of t	Attack Model	[105]	Our Result
Odd ($t = 2p - 1$)	KPA	$\sqrt{2} \cdot 2^{(p-\frac{k}{2})n} \cdot 2^{\frac{n}{2}}$	$\sqrt{2} \cdot 2^{(p-\frac{k}{2})n} \cdot 2^{-n}$
	CPA/CCA	$\sqrt{2} \cdot 2^{(p-\frac{k}{2})n} \cdot 2^{\frac{n}{2}}$	$\sqrt{2} \cdot 2^{(p-\frac{k}{2})n} \cdot 2^{-2n}$
Even ($t = 2p$)	KPA	$\sqrt{2} \cdot 2^{(p-\frac{k}{2})n}$	$\sqrt{2} \cdot 2^{(p-\frac{k}{2})n}$
	CPA/CCA	$\sqrt{2} \cdot 2^{(p-\frac{k}{2})n}$	$\sqrt{2} \cdot 2^{(p-\frac{k}{2})n} \cdot 2^{-n}$

3.8 Unbalanced Feistel Scheme

Figure 3.4: Round s of Unbalanced Feistel Scheme

3.8.1 Analysis of Unbalanced Feistel Scheme

In this section we only consider two special cases of $\text{UFN}(k', k)$. In Section 3.8.1.1, we discuss the analysis of the case when k is divisible by k' . This is a generalization of the UFN discussed in Section 6 of [115] where k is set to be 3 and k' is set to be 1. It can also be seen as a generalization of the UFN discussed in [116] where $k' = 1$. In Subsection 3.8.2, the case when $k - k'$ is a factor of k is considered. Due to the similarity of the technique used and also the page restriction, the detail of the analysis is omitted. This second case is a generalization of the analysis of $\text{UFN}(k', k)$ when $k' = k - 1$ in [81, 117, 131].

3.8.1.1 Analysis of UFN(k', k) when k' divides k

Let A be a positive integer such that $k = Ak'$. Define intermediate variables X_i such that $(X_{sk}, \dots, X_{sk+(k-1)})$ is the state value after s rounds. So

$$(X_0, \dots, X_{k-1}) = (S_0, \dots, S_{k-1}).$$

Suppose that the number of rounds is $r = pA + q$ where $0 \leq q \leq A - 1$. So $(X_{rk}, \dots, X_{rk+k-1}) = (I_0, \dots, I_{k-1})$. Given the state value after $s - 1$ ($s \geq 1$) backward rounds $(X_{(s-1)k}, \dots, X_{(s-1)k+k-1})$, the output of the s -th backward round can be computed by:

- $X_{sk+t} = X_{(s-1)k+(t-k')}$ if $k' \leq t \leq k - 1$,
- $X_{sk+t} = X_{(s-1)k+(k-k'+t)} \oplus F_{r+1-s,t}(X_{(s-1)k}, \dots, X_{(s-1)k+k-k'-1}) = X_{(s-1)k+(k-k'+t)} \oplus F_{r+1-s,t}(X_{sk+k'}, \dots, X_{sk+k-1})$ if $0 \leq t \leq k' - 1$.

Now for $0 \leq t \leq k' - 1$, expanding the relation given above, we get:

– If $q = 0$,

$$I_t = S_t \oplus \bigoplus_{s=1}^{p-1} F_{sA+2,t}(X_{(r-sA)k+k'}, \dots, X_{(r-sA)k+k-1}) \oplus F_{2,t}(I_{k'}, \dots, I_{k-1}),$$

– Otherwise,

$$I_t = S_{(A-q)k'+t} \oplus \bigoplus_{s=1}^p F_{sA+2,t}(X_{(r-sA)k+k'}, \dots, X_{(r-sA)k+k-1}) \oplus F_{2,t}(I_{k'}, \dots, I_{k-1}).$$

The distinguisher counts the number of set of two plaintext ciphertext pairs $(i, j), 1 \leq i < j \leq m$ such that

$$\forall t = 0, \dots, k' - 1, I_t(i) \oplus I_t(j) = S_{(a-q)k'+t}(i) \oplus S_{(a-q)k'+t}(j).$$

In CPA model, pick ciphertexts with a fixed value in $I_{k'}, \dots, I_{k-1}$. In other words, the maximum number of plaintext-ciphertext pairs is $m \leq 2^{k'n}$.

The expected values and variances of the random variables can be found in Table 3.5

Table 3.5: Expected Value and Variance for various cases of $\text{UFN}(k', k)$

Attack Model	q value	Π	E	V	σ
KPA	0	Perm	$\frac{m^2}{2 \cdot 2^{k'n}}$	$\frac{m^2}{2 \cdot 2^{k'n}}$	$\frac{m}{\sqrt{2} \cdot 2^{\frac{k'}{2}n}}$
		$\mathcal{F}$	$\frac{m^2}{2} \cdot \left(\frac{1}{2^{k'n}} + \frac{k'}{2^{(k'+p-1)n}} \right)$	$\frac{m^2}{2 \cdot 2^{k'n}}$	$\frac{m}{\sqrt{2} \cdot 2^{\frac{k'}{2}n}}$
	$1 \leq q \leq A-1$	Perm	$\frac{m^2}{2 \cdot 2^{k'n}}$	$\frac{m^2}{2 \cdot 2^{k'n}}$	$\frac{m}{\sqrt{2} \cdot 2^{\frac{k'}{2}n}}$
		$\mathcal{F}$	$\frac{m^2}{2} \cdot \left(\frac{1}{2^{k'n}} + \frac{k'}{2^{(k'+p)n}} \right)$	$\frac{m^2}{2 \cdot 2^{k'n}}$	$\frac{m}{\sqrt{2} \cdot 2^{\frac{k'}{2}n}}$
CPA	0	Perm	$\frac{m^2}{2 \cdot 2^{k'n}}$	$\frac{m^2}{2 \cdot 2^{k'n}}$	$\frac{m}{\sqrt{2} \cdot 2^{\frac{k'}{2}n}}$
		$\mathcal{F}$	$\frac{m^2}{2} \cdot \left(\frac{1}{2^{k'n}} + \frac{k'}{2^{(k'+p-2)n}} \right)$	$\frac{m^2}{2 \cdot 2^{k'n}}$	$\frac{m}{\sqrt{2} \cdot 2^{\frac{k'}{2}n}}$
	$1 \leq q \leq A-1$	Perm	$\frac{m^2}{2 \cdot 2^{k'n}}$	$\frac{m^2}{2 \cdot 2^{k'n}}$	$\frac{m}{\sqrt{2} \cdot 2^{\frac{k'}{2}n}}$
		$\mathcal{F}$	$\frac{m^2}{2} \cdot \left(\frac{1}{2^{k'n}} + \frac{k'}{2^{(k'+p-1)n}} \right)$	$\frac{m^2}{2 \cdot 2^{k'n}}$	$\frac{m}{\sqrt{2} \cdot 2^{\frac{k'}{2}n}}$

Using the definition of distinguishable, the complexity and maximum number of rounds distinguishable are summarised in Table 3.6 and 3.7.

A distinguisher for backward direction $\text{UFN}(k', k)$ can be constructed by considering the forward propagation of the equation. Hence, given the value of $X_{sk}, \dots, X_{sk+k-1}$, we have:

- $X_{sk+t} = X_{(s+1)k+(t+k')}$ if $0 \leq t \leq k - k' - 1$,
- $X_{sk+t} = X_{(s+1)k+(t-k+k')} \oplus F_{r-s,t}(X_{(s+1)k+k'}, \dots, X_{(s+1)k+k-1}) = X_{(s+1)k+(t-k+k')} \oplus F_{r-s,t}(X_{sk}, \dots, X_{sk+k-k'-1})$ if $k - k' \leq t \leq k - 1$.

Expanding S_t , the following equalities can be obtained:

Table 3.6: Complexity of Unbalanced Feistel Scheme

k'	q value	Attack Model	Complexity of distinguishing $pA + q$ rounds
1	0	KPA	$\sqrt{2} \cdot 2^{(p-\frac{1}{2})n}$
		CPA/CCA	$\sqrt{2} \cdot 2^{(p-\frac{3}{2})n}$
	$1 \leq q \leq A-1$	KPA	$\sqrt{2} \cdot 2^{(p+\frac{1}{2})n}$
		CPA/CCA	$\sqrt{2} \cdot 2^{(p-\frac{1}{2})n}$
$k' \geq 2$	0	KPA	$\frac{\sqrt{2}}{k'} \cdot 2^{(\frac{k'}{2}+p-1)n}$
		CPA/CCA	$\frac{\sqrt{2}}{k'} \cdot 2^{(\frac{k'}{2}+p-2)n}$
	$1 \leq q \leq A-1$	KPA	$\frac{\sqrt{2}}{k'} \cdot 2^{(p+\frac{k'}{2})n}$
		CPA/CCA	$\frac{\sqrt{2}}{k'} \cdot 2^{(p+\frac{k'}{2}-1)n}$

- If $q = 0$,

$$S_t = I_t \oplus \bigoplus_{s=1}^{p-1} F_{r-sA,t}(X_{sAk}, \dots, X_{sAk+k-k'-1}) \oplus F_{r,t}(S_0, \dots, I_{k-k'-1}),$$

- Otherwise,

$$S_t = I_{t-(A-q)k'} \oplus \bigoplus_{s=1}^p F_{r-sA,t}(X_{sAk}, \dots, X_{sAk+k-k'-1})$$

$$\oplus F_{r,t}(S_0, \dots, S_{k-k'-1}).$$

The distinguisher finds the number of sets of two plaintext-ciphertext pairs (i, j) such that $\forall t = k - k', \dots, k - 1, S_t(i) \oplus S_t(j) = I_{t-(A-q)k'}(i) \oplus I_{t-(A-q)k'}(j)$ where $S_0, \dots, S_{k-k'-1}$ are fixed in the CCA model. It is easy to see that with this model, we have exactly the same expected values, variances and distinguishability as the ones found in Table 3.5, 3.6 and 3.7.

Table 3.7: Summary of Distinguishability of Unbalanced Feistel Scheme

k'	q value	Attack Model	Maximum p distinguishable → Maximum round distinguishable
1	0	KPA	$k \rightarrow k^2$
		CPA/CCA	$2 \rightarrow 2k$
	$1 \leq q \leq A-1$	KPA	$k \rightarrow k^2 + k - 1$
		CPA/CCA	$1 \rightarrow k + k - 1$
$k' \geq 2$	0	KPA	$\begin{cases} k - \frac{k'}{2} + 1 \rightarrow \frac{k^2}{k'} - \frac{k}{2} + \frac{k}{k'} & \text{if } k' \text{ is even} \\ k - \frac{k'-1}{2} \rightarrow \frac{k^2}{k'} - \frac{k(k'-1)}{2k'} & \text{if } k' \text{ is odd} \end{cases}$
		CPA/CCA	$\begin{cases} \frac{k'}{2} + 2 \rightarrow \frac{k}{2} + 2\frac{k}{k'} & \text{if } k' \text{ is even} \\ \frac{k'+3}{2} \rightarrow \frac{k(k'+3)}{2k'} & \text{if } k' \text{ is odd} \end{cases}$
	$1 \leq q \leq A-1$	KPA	$\begin{cases} k - \frac{k'}{2} \rightarrow \frac{k^2}{k'} - \frac{k}{2} + \frac{k}{k'} - 1 & \text{if } k' \text{ is even} \\ k - \frac{k'+1}{2} \rightarrow \frac{k^2}{k'} - \frac{k(k'+1)}{2k'} + \frac{k}{k'} - 1 & \text{if } k' \text{ is odd} \end{cases}$
		CPA/CCA	$\begin{cases} \frac{k'}{2} + 1 \rightarrow \frac{k}{2} + \frac{k}{k'} + \frac{k}{k'} - 1 & \text{if } k' \text{ is even} \\ \frac{k'+1}{2} \rightarrow \frac{k(k'+1)}{2k'} + \frac{k}{k'} - 1 & \text{if } k' \text{ is odd} \end{cases}$

3.8.2 Result of Distinguishability of $\text{UFN}(k', k)$ when $(k - k')$ divides k

Let A be a positive integer such that $k = A(k - k')$. Since the case when $k'|k$ is already considered in Section 3.8.1.1, in this section, we assume that $k' \nmid k$. Because of this assumption, we have that $A \geq 3$. For the analysis, we consider the distinguishability of $r = pA + q$ rounds for some non-negative integers p, q such that $0 \leq q \leq A - 1$. Depending on the value of q , there are 5 cases to be considered. The analysis can be done using the same technique as all the previous sections. Due to the page restriction, in this section, we only present the complexity of distinguishing r rounds and the maximum value of p for various values of q such that distinguishing $r = pA + q$ rounds is still possible. As we have discussed in Section 3.8.1.1, in this section, we consider distinguisher in KPA and CPA model. Distinguisher for CCA model can

similarly be constructed using the same analysis.

1. $q = 0$.

- KPA:

- Complexity of distinguishing pA rounds for any non-negative integer p is $\frac{\sqrt{2}}{k'} \cdot 2^{(\frac{k'}{2} + p(A-1)-1)n}$.
- Maximum value of p such that the complexity of distinguishing pA rounds is less than 2^{kn} : $p_{\max} = \left\lfloor \frac{k - \frac{k'}{2} + 1}{A-1} \right\rfloor$.

- CPA: Our distinguisher and choice of plaintexts require the complexity to be at most $2^{k'n}$.

- Complexity of distinguishing pA rounds for any non-negative integer p is $\frac{\sqrt{2}}{k'} \cdot 2^{(\frac{k'}{2} + p(A-1)-2)n}$.
- Maximum value of p such that the complexity of distinguishing pA rounds is less than $2^{k'n}$: $p_{\max} = \left\lfloor \frac{\frac{k'}{2} + 2}{A-1} \right\rfloor$.

2. $q = 1$. In this case, we further divide the case to $A = 3$ and $A \geq 4$.

- $A = 3$.

- KPA:

- * Complexity of distinguishing $3p + 1$ rounds for any non-negative integer p is $\frac{\sqrt{2}}{\lfloor \frac{k-k'}{2} \rfloor} \cdot 2^{(\frac{k'}{2} + 2p-2)n}$.
- * Maximum value of p such that the complexity of distinguishing $3p + 1$ rounds is at most 2^{kn} :

$$p_{\max} = \left\lfloor \frac{k}{2} - \frac{k'}{4} + 1 + \frac{1}{2n} \left(\log \left\lfloor \frac{k - k'}{2} \right\rfloor - \frac{1}{2} \right) \right\rfloor.$$

- CPA: Our distinguisher and choice of plaintexts require the complexity to be at most $2^{k'n}$.

- * Complexity of distinguishing $3p + 1$ rounds for any non-negative integer p is $\frac{\sqrt{2}}{\lfloor \frac{k-k'}{2} \rfloor} \cdot 2^{(\frac{k'}{2} + 2p-3)n}$.

- * Maximum value of p such that the complexity of distinguishing $3p + 1$ rounds is at most $2^{k'n}$:

$$p_{\max} = \left\lfloor \frac{k'}{4} + \frac{3}{2} + \frac{1}{2n} \left(\log \left\lfloor \frac{k - k'}{2} \right\rfloor - \frac{1}{2} \right) \right\rfloor.$$

- $A \geq 4$.

– KPA:

- * Complexity of distinguishing $pA + 1$ rounds for any non-negative integer p is $\frac{\sqrt{2}}{k' - \lceil \frac{k}{A-1} \rceil} \cdot 2^{(\frac{k'}{2} + p(A-1) - 2)n}$.
- * Maximum value of p such that the complexity of distinguishing $pA + 1$ rounds is at most 2^{kn} :

$$p_{\max} = \left\lfloor \frac{1}{A-1} \cdot \left(k - \frac{k'}{2} + 2 + \frac{1}{n} \left(\log \left(k' - \left\lceil \frac{k}{A-1} \right\rceil \right) \right) - \frac{1}{2} \right) \right\rfloor.$$

– CPA: Our distinguisher and choice of plaintexts require the complexity to be at most $2^{(A-2)(k-k')n}$.

- * Complexity of distinguishing $pA + 1$ rounds for any non-negative integer p is $\frac{\sqrt{2}}{k' - \lceil \frac{k}{A-1} \rceil} \cdot 2^{(\frac{(A-2)(k-k')}{2} + p(A-1) - 3)n}$.
- * Maximum value of p such that the complexity of distinguishing $pA + 1$ rounds is at most $2^{(A-2)(k-k')n}$:

$$p_{\max} = \left\lfloor \frac{1}{A-1} \cdot \left(k - \frac{(A-2)(k-k')}{2} + 3 + \frac{1}{n} \left(\log \left(k' - \left\lceil \frac{k}{A-1} \right\rceil \right) \right) - \frac{1}{2} \right) \right\rfloor.$$

$$3. \ 2 \leq q \leq \left\lfloor \frac{(A-1)(k'-1)}{k} \right\rfloor.$$

- KPA:

– Complexity of distinguishing $pA + q$ rounds for any non-negative integer p is $\frac{\sqrt{2}}{k' - q(k-k') - \lceil \frac{q(k-k')}{A-1} \rceil} \cdot 2^{(\frac{k'}{2} + p(A-1) - q - 1)n}$.

- Maximum value of p such that the complexity of distinguishing $pA + q$ rounds is at most 2^{kn} :

$$p_{\max} = \left\lfloor \frac{1}{A-1} \left(k - \frac{k'}{2} + q + 1 + \frac{1}{n} (\log(k' - q(k - k')) - \left\lceil \frac{q(k - k')}{A-1} \right\rceil) - \frac{1}{2} \right) \right\rfloor.$$

- CPA: Our distinguisher and choice of plaintexts require the complexity to be at most $2^{(\lfloor (A-1 - \frac{q}{A-1})(k-k') \rfloor)n}$.

- Complexity of distinguishing $pA + q$ rounds for any non-negative integer p is $\frac{\sqrt{2}}{k' - q(k - k') - \lceil \frac{q(k - k')}{A-1} \rceil} \cdot 2^{(\frac{1}{2}(k' - \lceil \frac{q(k - k')}{A-1} \rceil) + p(A-1) - q - 2)n}$.
- Maximum value of p such that the complexity of distinguishing $pA + q$ rounds is at most $2^{(\lfloor (A-1 - \frac{q}{A-1})(k-k') \rfloor)n}$:

$$p_{\max} = \left\lfloor \frac{1}{A-1} \left(\frac{1}{2} \left(k' - \left\lceil \frac{q(k - k')}{A-1} \right\rceil \right) + q + 2 + \frac{1}{n} \left(\log \left(k' - q(k - k') - \left\lceil \frac{q(k - k')}{A-1} \right\rceil \right) - \frac{1}{2} \right) \right) \right\rfloor.$$

$$4. \left\lceil \frac{(A-1)(k'-1)+1}{k} \right\rceil \leq q \leq \left\lfloor \frac{(A-1)(k'-1)}{k} \right\rfloor + 1.$$

- KPA:
 - Complexity of distinguishing $pA + q$ rounds for any non-negative integer p is $\frac{\sqrt{2}}{k'} \cdot 2^{(\frac{k'}{2} + p(A-1) - q)n}$.
 - Maximum value of p such that the complexity of distinguishing $pA + q$ rounds is at most 2^{kn} :

$$p_{\max} = \left\lfloor \frac{1}{A-1} \left(k - \frac{k'}{2} + q + \frac{1}{n} \left(\log k' - \frac{1}{2} \right) \right) \right\rfloor$$

- CPA: Our distinguisher and choice of plaintexts require the complexity to be at most $2^{k'n}$.
 - Complexity of distinguishing $pA + q$ rounds for any non-negative integer p is $\frac{\sqrt{2}}{k'} \cdot 2^{(\frac{k'}{2} - 1 + p(A-1) - q)n}$.

- Maximum value of p such that the complexity of distinguishing $pA + q$ rounds is at most $2^{k'n}$:

$$p_{\max} = \left\lfloor \frac{1}{A-1} \left(\frac{k'}{2} + q + 1 + \frac{1}{n} \left(\log k' - \frac{1}{2} \right) \right) \right\rfloor.$$

$$5. \left\lceil \frac{(k'-1)(A-1)+1}{k} \right\rceil + 1 \leq q \leq A-1.$$

- KPA:

- Complexity of distinguishing $pA + q$ rounds for any non-negative integer p is $\frac{\sqrt{2}}{k'} \cdot 2^{(\frac{k'}{2} + p(A-1) - q)n}$.
- Maximum value of p such that the complexity of distinguishing $pA + q$ rounds is at most 2^{kn} :

$$p_{\max} = \left\lfloor \frac{1}{A-1} \left(k - \frac{k'}{2} + q + \frac{1}{n} \left(\log k' - \frac{1}{2} \right) \right) \right\rfloor.$$

- CPA: Our distinguisher and choice of plaintexts require the complexity to be at most $2^{k'n}$.

- Complexity of distinguishing $pA + q$ rounds for any non-negative integer p is $\frac{\sqrt{2}}{k'} \cdot 2^{(\frac{k'}{2} + p(A-1) - q - 1)n}$.
- Maximum value of p such that the complexity of distinguishing $pA + q$ rounds is at most $2^{k'n}$:

$$p_{\max} = \left\lfloor \frac{1}{A-1} \left(\frac{k'}{2} + q + 1 + \frac{1}{n} \left(\log k' - \frac{1}{2} \right) \right) \right\rfloor.$$

3.9 Conclusion

In this chapter, differential analysis on the inverse function of four different types of generic Generalized Feistel Scheme, namely Type-1, Type-3, Alternating Scheme and UFN(k', k) was considered. We show that for Type-1 Feistel Scheme, backward distinguisher performs better especially in the chosen ciphertext attack compared to the results in [105]. Using the same complexity, we can distinguish $k - 1$ more rounds

while distinguishing the same number of rounds requires smaller complexity with factor of $\frac{1}{2^n}$.

In Type-2 and Alternating Feistel scheme, although there are some difference in the complexity, both directions can achieve almost the same number of rounds. This shows that these two types can be seen as almost symmetric from both direction.

We improve the differential cryptanalysis in Type-3 Feistel Scheme in several cases. In KPA model with low number of branches, $2 \leq k \leq 5$, our analysis provides a lower bound of the number of rounds that is indistinguishable from random permutation. Secondly, in CCA model, the lower bound of maximum number of rounds distinguishable is increased by 1 round, from $k + 1$ obtained in [105] to $k + 2$.

In Alternating Feistel Scheme, we achieve 2 more rounds than the one claimed in [105]. The complexity is reduced by a factor of $2^{\frac{3n}{2}}$ when distinguishing the same odd number of rounds.

Lastly, a lower bound for the maximum number of rounds that is distinguishable from random permutation in $\text{UFN}(k', k)$ scheme is given through the forward direction distinguisher. To the best of our knowledge, this is the first bound given in a rather general case in which k' is arbitrary as long as k' is a divisor of k for any integer k .

Chapter 4

Differential-Linear Cryptanalysis of ICEPOLE

4.1 Introduction

In the Early Symmetric Crypto (ESC) 2013, CAESAR competition has been announced to find a good design of authenticated encryption scheme. The criteria for the designs are security against various known attacks as well as its performance.

The call for submission results in 55 designs advancing to the first round. In July 2015, from these 55 designs, 28 designs advanced to the second round. In August 2016, 15 designs were announced to advance to the third round which is still under review until now.

ICEPOLE is a new hardware-oriented single-pass authenticated cipher designed by Morawiecki et al. It was submitted to the CAESAR competition [102] and published in CHES 2014 [103]. ICEPOLE is designed to be hardware-efficient. It can achieve 41 Gbits/s on the modern FPGA device Virtex 6 which is over 10 times faster than the equivalent implementation of AES-128-GCM [98]. ICEPOLE adopts the well-known duplex construction by Bertoni et al. [22] and uses a Keccak-like permutation as its iterative function.

The ICEPOLE family of authenticated ciphers includes three variants: ICEPOLE-128, ICEPOLE-128a and ICEPOLE-256a. Note that the definition of ICEPOLE-128 has been slightly modified in the CHES 2014 version, which removed the use of secret message number. In this chapter, we will follow the version submitted to the CAESAR competition. For the security of ICEPOLE, the designers claim that the confidentiality is the same as the key length, which is 128-bit for ICEPOLE-128 and ICEPOLE-128a , and 256-bit for ICEPOLE-256a. The authentication security is 128-bit for all the three variants. In particular, it is mentioned in the document that the internal permutation is strong enough such that “*in the case of nonce reuse the key-recovery attack is not possible*” and the authenticity “*is not threatened by nonce reuse*”.

In this chapter, we apply the differential-linear cryptanalysis to study the security of the permutation of the ICEPOLE family. The differential-linear cryptanalysis is the combination of differential cryptanalysis [29] and linear cryptanalysis [96]. It was introduced by Langford and Hellman in [90] in 1994 to attack the block cipher DES, and later Biham, Dunkelman and Keller gave an enhanced version of this method [26]. This method has been applied to analyze a number of block ciphers such as Serpent [27, 56], CTC2 [58, 93] and SHACAL-2 [126]. Differential-linear attack is also successful in the analysis of certain stream ciphers, *e.g.*, Phelix which involves the message in the state update function [142].

Although the design of authenticated cipher ICEPOLE is different from block ciphers, we manage to exploit the differential-linear property of the permutation when the nonce is reused ¹. We show that under the nonce-reuse assumption, there exists distinguishing attacks on ICEPOLE with both time and data complexity less than 2^{36} . Furthermore, it is possible to recover the 256 bits unknown state of ICEPOLE-128 and ICEPOLE-128a with practical complexity 2^{46} , and recover the 320 bits unknown

¹Here *nonce* includes the public message number and secret message number. The associated data is set to be identical or empty which is generally allowed.

state of ICEPOLE-256a with complexity 2^{60} . We experimentally verified our results by recovering the state of ICEPOLE-128 using a 64-core server within 10 days. Thus, the security claims of ICEPOLE do not hold under the nonce-reuse circumstances.

Due to the analysis here, the designers have updated the security claims as “*in the case of nonce misuse, the intermediate level of robustness (specified in the documentation) holds only when the SMN is present and respected, namely each message has the corresponding, unique secret message number*” [101]. In the presence of unique SMN, this attack will no longer work. The reason is that the unique SMN plays the role of the nonce in the initialization, and prevents the differential attack in the message processing.

The rest of this chapter is structured as follows: The specification of ICEPOLE is given in Section 4.2. Section 4.3 describes a differential-linear distinguishing attack on ICEPOLE. Section 4.4 introduces the state-recovery attack. Section 4.5 provides our experimental results of the state-recovery attack on ICEPOLE-128. Section 4.6 concludes the chapter.

4.2 The ICEPOLE Authenticated Cipher

The ICEPOLE family of authenticated ciphers uses three parameters: key length (128 or 256 bits), secret message number (SMN) length (0 or 128 bits) and nonce length (96 or 128 bits). ICEPOLE-128 has 128-bit secret message number, 128-bit key, and 128-bit nonce. The other two variants, ICEPOLE-128a and ICEPOLE-256a, have no secret message number with 128- and 256-bit secret key respectively. The nonce length for these two variants is 96-bit. We will briefly describe the specification of ICEPOLE authenticated cipher. The full specification can be found in [102]. An overview of ICEPOLE-128 is provided in Figure 4.1.

Figure 4.1: General scheme of ICEPOLE encryption and authentication (Fig. 1 of [102])

4.2.1 Notations

The ICEPOLE algorithm has a 1280-bit internal state S . It uses the little-endian convention. The organization of internal state is similar to Keccak [21], which uses a 3-dimension structure. Therefore, the 1280-bit state S can be represented as $S[4][5][64]$, or shortly $S[4][5]$, an array of 64-bit words. For $S[x][y][z]$, it is corresponding to the $64(x + 4y) + z$ -th bit of the input. ICEPOLE uses 4×5 slices. Each slice has 4 rows and 5 columns. And $S_{[n]}$ denotes the first n bits of the state.

4.2.2 The ICEPOLE permutation P

The permutation P is applied iteratively on the ICEPOLE state S during the encryption and authentication. Each permutation is called a *round* or R . The 6- and 12-round of P are represented as P_6 and P_{12} respectively. Each round includes five operations: μ , ρ , π , ψ , and κ .

$$R = \kappa \circ \psi \circ \pi \circ \rho \circ \mu$$

The operations are defined as follows:

μ :

A column vector (Z_0, Z_1, Z_2, Z_3) is multiplied by a constant matrix to produce a

vector of four 5-bit words.

$$\begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 1 & 18 & 2 \\ 1 & 2 & 1 & 18 \\ 1 & 18 & 2 & 1 \end{bmatrix} \times \begin{bmatrix} Z_0 \\ Z_1 \\ Z_2 \\ Z_3 \end{bmatrix} = \begin{bmatrix} 2Z_0 + Z_1 + Z_2 + Z_3 \\ Z_0 + Z_1 + 18Z_2 + 2Z_3 \\ Z_0 + 2Z_1 + Z_2 + 18Z_3 \\ Z_0 + 18Z_1 + 2Z_2 + Z_3 \end{bmatrix}$$

The operations are done in $\mathbb{F}_{2^5} \cong \mathbb{F}_2[x]/(x^5 + x^2 + 1)$. And μ can be efficiently implemented with simple bitwise equations, see Appendix B in [102].

ρ :

The ρ step is the bitwise rotation on each of the 20 64-bit words. It is defined as:

$$\rho(S[x][y]) = S[x][y] \lll \text{offsets}[x][y] \quad \text{for all } (0 \leq x \leq 3), (0 \leq y \leq 4)$$

The rotation offsets are as follows:

$$\begin{array}{llll} \text{offset}[0][0] := 0 & \text{offset}[0][1] := 36 & \text{offset}[0][2] := 3 & \text{offset}[0][3] := 41 \\ \text{offset}[0][4] := 18 & \text{offset}[1][0] := 1 & \text{offset}[1][1] := 44 & \text{offset}[1][2] := 10 \\ \text{offset}[1][3] := 45 & \text{offset}[1][4] := 2 & \text{offset}[2][0] := 62 & \text{offset}[2][1] := 6 \\ \text{offset}[2][2] := 43 & \text{offset}[2][3] := 15 & \text{offset}[2][4] := 61 & \text{offset}[3][0] := 28 \\ \text{offset}[3][1] := 55 & \text{offset}[3][2] := 25 & \text{offset}[3][3] := 21 & \text{offset}[3][4] := 56 \end{array}$$

π :

π reorders the bits within each slice. It maps $S[x][y]$ to $S[x'][y']$ using following rule:

- $x' := (x + y) \bmod 4$
- $y' := (((x + y) \bmod 4) + y + 1) \bmod 5$

ϕ :

ϕ is the S-box layer. ICEPOLE uses following 5-bit S-box:

$$\{ 31, 9, 18, 11, 5, 12, 22, 15, 10, 3, 24, 1, 13, 4, 30, 7, \\ 20, 21, 6, 23, 17, 16, 2, 19, 26, 27, 8, 25, 29, 28, 14, 0 \}$$

The ϕ applies the 5-bit S-box to all the 4 rows of the 64 slices of the state.

κ :

In κ the 64-bit constant is XORed with $S[0][0]$. The constants are different for each round and we omit the values here.

4.2.3 Initialization

First, the state S is initialized with 1280-bit constant:

$$\begin{array}{ll} S[0][0] := 0xFF97A42D7F8E6FD4 & S[0][1] := 0x90FEE5A0A44647C4 \\ S[0][2] := 0x8C5BDA0CD6192E76 & S[0][3] := 0xAD30A6F71B19059C \\ S[0][4] := 0x30935AB7D08FFC64 & S[1][0] := 0xEB5AA93F2317D635 \\ S[1][1] := 0xA9A6E6260D712103 & S[1][2] := 0x81A57C16DBCF\ 555F \\ S[1][3] := 0x43B831CD0347C826 & S[1][4] := 0x01F22F1A11A5569F \\ S[2][0] := 0x05E5635A21D9AE61 & S[2][1] := 0x64BEFEF28CC970F2 \\ S[2][2] := 0x613670957BC46611 & S[2][3] := 0xB87C5A554FD00ECB \\ S[2][4] := 0x8C3EE88A1CCF32C8 & S[3][0] := 0x940C7922AE3A2614 \\ S[3][1] := 0x1841F924A2C509E4 & S[3][2] := 0x16F53526E70465C2 \\ S[3][3] := 0x75F644E97F30A13B & S[3][4] := 0xEAF1FF7B5CECA249 \end{array}$$

Then, the key (K) and the *nonce* are XORed to the state. The *nonce* is 128-bit for ICEPOLE-128. And the 96-bit *nonce* for ICEPOLE-128a and ICEPOLE-256a will be padded with 32 zeros to form a 128-bit *nonce*. $nonce_0$ and $nonce_1$ denote two 64-bit words of the padded nonce.

For ICEPOLE-128 and ICEPOLE-128a, K_0 and K_1 denote two 64-bit words of the key,

$$\begin{aligned}
S[0][0] &:= S[0][0] \oplus K_0 \\
S[1][0] &:= S[1][0] \oplus K_1 \\
S[2][0] &:= S[2][0] \oplus \textit{nonce}_0 \\
S[3][0] &:= S[3][0] \oplus \textit{nonce}_1
\end{aligned}$$

For ICEPOLE–256a, K_0 , K_1 , K_2 and K_3 denote four 64-bit words of the key,

$$\begin{aligned}
S[0][0] &:= S[0][0] \oplus K_0 \\
S[1][0] &:= S[1][0] \oplus K_1 \\
S[2][0] &:= S[2][0] \oplus K_2 \\
S[3][0] &:= S[3][0] \oplus K_3 \\
S[0][1] &:= S[0][1] \oplus \textit{nonce}_0 \\
S[1][1] &:= S[1][1] \oplus \textit{nonce}_1
\end{aligned}$$

After that, the P_{12} permutation is applied to the state S .

4.2.4 Processing associated data and plaintext

ICEPOLE–128 uses 128-bit secret message number (SMN) σ^{SMN} . It will be processed before associated data and the plaintext. Since ICEPOLE–128a and ICEPOLE–256a do not have SMN, only the associated data σ_i^{AD} and the plaintext σ_i^P will be processed.

For ICEPOLE–128 and ICEPOLE–128a, the length of blocks σ_i^{AD} and σ_i^P is in the range $[0, 1024]$ bits. The blocks will be padded to 1026 bits according to following rule. First, a *frame bit* is appended. It is set to ‘1’ for the last σ^{AD} block and all σ_i^P except the last one. For all other blocks, it is set to ‘0’. Then, a bit ‘1’ is appended,

following by '0's to make the length of padded block be 1026-bit. The number of blocks under a single key is less than 2^{126} .

For ICEPOLE-256a, the associated data and plaintext blocks have length in the range $[0, 960]$. The same padding rule is applied and the padded blocks have length 962-bit. The number of blocks under a single key is less than 2^{62} .

The process of secret message number is as below for ICEPOLE-128:

$$c_{SMN} = S_{\lfloor 128 \rfloor} \oplus \sigma^{SMN}$$

$$\sigma^{SMN} := pad(\sigma^{SMN})$$

$$S_{\lfloor 1026 \rfloor} := S_{\lfloor 1026 \rfloor} \oplus \sigma^{SMN}$$

$$S := P_6(S)$$

The process of associated data and plaintext blocks is as below:

for all blocks $\sigma_i^{AD}\{$

$$\sigma_i^{AD} := pad(\sigma_i^{AD})$$

$$S_{\lfloor 1026 \rfloor} := S_{\lfloor 1026 \rfloor} \oplus \sigma_i^{AD}$$

$$S := P_6(S)$$

$\}$

for all blocks $\sigma_i^P\{$

$$c_i := S_{\lfloor l \rfloor} \oplus \sigma_i^P \text{ (} l \text{ is the length of } \sigma_i^P \text{)}$$

$$\sigma_i^P := pad(\sigma_i^P)$$

$$S_{\lfloor 1026 \rfloor} := S_{\lfloor 1026 \rfloor} \oplus \sigma_i^P$$

$$S := P_6(S)$$

$\}$

4.2.5 Tag generation

After the AD and P are processed, the 128-bit tag T is derived: (T_0 and T_1 are two 64-bit words of T).

$$T_0 := S[0][0]$$

$$T_1 := S[0][1]$$

The decryption and verification is trivial and we omit it here.

4.2.6 Security goals of ICEPOLE

The main security goals of ICEPOLE are: 128-bit encryption security for ICEPOLE-128 and ICEPOLE-128a; 256-bit encryption security for ICEPOLE-256a; and 128-bit authentication security for all variants.

An important property of ICEPOLE is that the intermediate level of robustness under nonce-misuse circumstance. It claimed that

1. “...in the case of nonce reuse the key-recovery attack is not possible”.
2. “Authenticity (integrity) in the duplex construction does not need a nonce requirement, thus is not threatened by nonce reuse.

4.3 Differential-Linear Distinguishing Attack on ICEPOLE

In [102], the designers have performed initial cryptanalysis on ICEPOLE, including the differential cryptanalysis, linear cryptanalysis and rotational cryptanalysis. In this section, we will revisit the cryptanalysis in [102] and introduce our analysis on ICEPOLE based on the differential-linear cryptanalysis. We would like to emphasize that our attacks only work under the assumption that the *nonce* and *secret message number* can be reused.

Before we go on to the attack, we recall the main idea of Differential-Linear Cryptanalysis discussed in Subsection 1.2.3.

The main idea is to query messages with certain input difference and analyze the statistics of the differences of chosen bits (according to the linear mask) in the output. When the XORed differences of the chosen bits have a significant bias from 0.5, the adversary can distinguish the cipher from a random permutation. In [93], Lu studied the implicit assumptions made in [90] and [26] and gave a theorem to compute the probability for the differential-linear distinguisher under the original two assumptions:

1. The involved round functions behave independently.
2. The two inputs $E_0(P)$ and $E_0(P \oplus \alpha)$ of the linear characteristic for E_1 behave as independent inputs with respect to the linear characteristic, where E_0 is the encryption for the differential rounds and E_1 is the encryption for the linear rounds, P is the plaintext and α is the input difference.

Let $\hat{p}$ be the probability that the input of the linear mask has no XORed difference after the differential step while ϵ be the linear characteristic bias for the linear step. The theorem says that the probability that the XORed difference of the output linear mask to be 0 is $\frac{1}{2} + 2(2\hat{p} - 1)\epsilon^2$. In [40], Blondeau et al. further developed a method on computing the bias which only relies on the independence of the two parts of the cipher.

Hence, when the bias is large enough, it is possible to distinguish it from a random permutation. In the analysis on ICEPOLE, we first divide the encryption P_6 into two parts with equal number of rounds. So the first 3 rounds will be the differential step and the last 3 rounds will be the linear step.

Our task is to find good 3-round differential characteristics and 3-round linear characteristics. Unless otherwise specified, we are discussing the ICEPOLE-128 and ICEPOLE-128a in this section under the nonce-misuse assumption. The ICEPOLE-256a is similar and we will discuss it later.

4.3.1 Constructing the differential characteristics

In ICEPOLE, S-box is the only non-linear operation, and the maximum differential probability of the S-box is 2^{-2} . The differential probability is largely determined by the number of active S-boxes. Although the designers expected that only 3% of the difference transitions has the maximum probability 2^{-2} , this probability is not rare in the early rounds. This is because most of the active S-boxes have 1 bit input difference after the diffusion and the differential probability of a single ICEPOLE S-box is 2^{-2} when the input and output differences are identical with weight 1. Those 1-to-1 identical difference transitions are preferred to the 1-to- n ($n \geq 2$) cases even with the same probability as they will propagate to less number of active S-boxes in the next round.

In [102], the designers analyzed the minimum number of active S-boxes using SAT solver and found that the minimum number is 9 for 3 rounds ICEPOLE. There is an intuitive way to construct the differential characteristics reach this lower bound. We can place 1 active S-box in the middle round and let it propagate backward and forward for one round. Then the number of active S-boxes will be in the form of 4-1-4, which reaches the minimum number 9.

However, the above 3-round differential path is not feasible. In fact, only at most 1024 bits (for ICEPOLE-256a, 960 bits) out of the 1280 bits in a state can be affected by the plaintext and are feasible to introduce difference. Thus, we have the following observation on the operation μ .

Observation 4.3.1. *For any 20-bit slice, when the output difference is one bit after μ , there is at least one bit input difference at the last column ($S[\cdot][4]$).*

This observation can be easily verified by analyzing the inverse operation of μ .

Therefore, when an active S-box is propagated backward, it is likely that a number of active bits will be propagated to the last column of the input state, which is infeasible to introduce.

It implies that for any feasible differential characteristics of ICEPOLE, the minimum number of active S-boxes is 2 in the first round. As a result, we will consider the differential characteristics with two active S-boxes in the first round. The ideal case is that the output difference of each active S-box is only 1-bit. Then after μ , this 1-bit difference in a slice will propagate to 4 bits. So we expect the good differential characteristics will have 8 active S-boxes in round 2 and no more than 32 active S-boxes in round 3.

We searched for good 3-round differential characteristics, and found 5 possible initial differences in Table 4.1 (the rotated differences on the 64-bit word are not considered here), which can lead to feasible 3-round differential characteristics. We will name them as D_1 , D_2 , D_3 , D_4 and D_5 . Here the differentials are expressed as 4×5 matrices with each element being the difference in the corresponding position of the state, expressed as 64-bit string, representing the differences in the 64 slices in that position.

The total number of 3-round active S-boxes is 42, which implies that the differential probability is at most 2^{-84} . But in fact, the only requirement is that the input linear mask does not have XORed difference at the beginning of round 4. Hence, a large number of differential characteristics are satisfied. As long as the number of active S-boxes is relatively low (note that 32 is only 1/8 of the total 256 S-boxes), there is a high probability for the differential step.

4.3.2 Constructing the linear characteristic

The bias of an ICEPOLE linear characteristic is determined by the S-boxes involved in the linear characteristic. We use the following method to construct the 3-round linear characteristics.

Take a single bit in both the input and output masks of an S-box in the middle round. Then find related bits in the input of the first round (*input linear mask*) and the output of the third round before S-box (*output linear mask*), assuming that

Table 4.1: The initial differences. Each entry represents a 64-bit word.

(a) D_1					(b) D_2				
0x0	0x0	0x1	0x0	0x0	0x0	0x0	0x1	0x0	0x0
0x1	0x0	0x1	0x0	0x0	0x1	0x1	0x1	0x1	0x0
0x1	0x1	0x1	0x1	0x0	0x0	0x1	0x0	0x1	0x0
0x0	0x1	0x0	0x0	0x0	0x1	0x0	0x1	0x0	0x0

(c) D_3				
0x1	0x1	0x0	0x0	0x0
0x1	0x0	0x0	0x1	0x0
0x0	0x0	0x0	0x1	0x0
0x1	0x1	0x1	0x0	0x0

(d) D_4					(e) D_5				
0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
0x1	0x0	0x1	0x0	0x0	0x0	0x1	0x0	0x1	0x0
0x1	0x1	0x1	0x1	0x0	0x1	0x0	0x1	0x0	0x0
0x0	0x1	0x0	0x1	0x0	0x1	0x1	0x1	0x1	0x0

the S-boxes are identical mappings. The rationale here is that the 1-bit identical mappings have a bias $\frac{3}{16}$, it nearly reaches the maximum bias 2^{-2} while keeping the masks low-weight.

The most essential criterion for the input and output masks is low-weight. When the input linear mask has lower weight, the probability that the input linear mask does not have XORed difference after the 3-round differential step will be higher. When the output linear mask has lower weight, less number of active S-boxes in round 6 will be involved in the linear relation.

Two good linear characteristics we found are given in Table 4.2. They are denoted as L_1 and L_2 .

Table 4.2: The linear characteristics, assuming S-boxes are identical mappings

(a) Linear characteristic 1, input linear mask

0x0	0x8000000000000000	0x2000000000	0x20000	0x8000000000000040
0x40	0x0	0x8000000000000000	0x2000020000	0x0
0x0	0x2000000000	0x8000000000000000	0x40	0x2000020000
0x0	0x0	0x800002000020000	0x0	0x40

(b) Linear characteristic 1, output linear mask

0x40000	0x1	0x0	0x800000000000	0x0
0x0	0x4	0x0	0x0	0x0
0x0	0x200000	0x2000000000000000	0x0	0x0
0x0	0x0	0x200000000000	0x1000000000000000	0x0

(c) Linear characteristic 2, input linear mask

0x120000004000000	0x8000000000000000	0x100000000000	0x0	0x0
0x8020000000000000	0x4000000	0x8000100000000000	0x0	0x1000000000000000
0x8100000000000000	0x2000000000000000	0x0	0x100000000000	0x4000000
0x4000000	0x8100100000000000	0x0	0x0	0x2010000000000000

(d) Linear characteristic 2, output linear mask

0x40000	0x1	0x0	0x0	0x0
0x8000	0x4	0x0	0x0	0x0
0x0	0x0	0x2000000000000000	0x0	0x0
0x0	0x400	0x200000000000	0x1000000000000000	0x0

4.3.3 Observations to improve the attack

The following observation will be helpful in the selection of the differential characteristics.

Observation 4.3.2. *When the 1024 bits $S[0.3][0.3]$ of an ICEPOLE state S are known, we are able to determine $S[0][2]$, $S[1][3]$, $S[2][4]$, $S[3][0]$, $S[3][2]$ after μ , ρ and π operations are performed on S .*

According to the above observation, we are able to determine the values of certain input bits to the S-boxes in the first round. Thus, we can update the differential tables for the S-boxes with fixed input values.

The following example shows how this will help to improve the differential probability. Suppose the input of an S-box is in the form “1 * 0 * *”, where the ‘*’s are the unknown bits and the ‘1’ and ‘0’ are the bits with fixed values, and the input difference is 2, then the output difference is 2 with probability 1, which is larger than 0.25 in the general case. To get the fixed values, we can manipulate the input bits

according to the mask in Table 4.3.

Table 4.3: Input mask for the fixed bits in round 1.

0x0	0x800000010	0x200000001	0x0	0x0
0x800000010	0x0	0x0	0x200000001	0x0
0x0	0x800000010	0x0	0x200000001	0x0
0x800000010	0x0	0x800000010	0x200000001	0x0

By setting the bits in $S[0][1]$ selected by the mask 0x800000010 as ‘1’ and all the other bits selected by the other masks as ‘0’, the two active S-boxes in round 1 will satisfy the above condition on the input values.

Our next observation is about the S-box in the round 6 of P_6 .

Observation 4.3.3. *When 4 of the 5 bits in the output of an S-box are known, it is possible to recover some of the input bits from the output bits of the S-boxes. Table 4.4 provides the probability that we can recover a bit at each position of the S-box input.*

Since the ciphertext is generated by XORing the keystream and the plaintext, we can obtain the values of at most 1024 bits in the state after 6 rounds. For ICEPOLE-128 and ICEPOLE-128a, 4 of the 5 bits in the output of all the S-boxes are known. And for ICEPOLE-256a, 4 of the 5 bits for 192 S-boxes and 3 of the 5 bits for 64 S-boxes in the output are known.

Therefore, instead of using the bias of the linear relation in the round 6, we can recover the chosen input bits in round 6 before S-box by enumerating the values of output bits. This can then be used to recover the value of the bit in the linear relation in the output of round 5.

Table 4.4: Probability that the input bits can be recovered for an S-box at round 6.

Position	0	1	2	3	4
Probability	$\frac{6}{8}$	$\frac{5}{8}$	$\frac{4}{8}$	$\frac{4}{8}$	$\frac{1}{8}$

Table 4.5: The difference of state before the first S-box in D_2 .

0x0	0x0	0x0	0x0	0x0
0x0	0x0	0x0	0x0	0x0
0x0	0x0	0x0	0x0	0x0
0x0	0x400	0x200000000000	0x0	0x0

For example, if we consider the output linear mask of L_1 , the probability that we can recover the 8 bits can be computed as

$$Pr_{L_1} = (3/4) \times (5/8)^3 \times (1/2)^4 = 2^{-6.45}.$$

And using these 8 bits, we can compute the value of bit $S[1][1][0]$ at the output of round 5.

Similarly, the probability for L_2 is

$$Pr_{L_2} = (3/4)^2 \times (5/8)^3 \times (1/2)^3 = 2^{-5.86}.$$

4.3.4 Concatenating the differential and linear characteristics

After the 3-round differential characteristics and 3-round linear characteristics are constructed, we can concatenate them to form 6-round differential-linear characteristics.

First, we choose the initial difference D_2 because the two active S-boxes in round 1 are in the last row which has two known bits. The difference of state before the S-box for D_2 is given in Table 4.5.

To choose the 3-round linear characteristic, we consider all the possible rotations of the two linear characteristics L_1 and L_2 given in Section 4.3.2. There are 128 possible rotated linear characteristics. The selection of linear characteristic is done experimentally:

1. Randomly pick 1024-bit plaintext blocks pairs with the chosen initial difference D_2 and the fix values (1, 0) for the two known bits at positions 0 and 2 in the

round 1 active S-boxes.

2. After 5 rounds, verify whether the XORed difference of the states under the linear characteristic is zero or not. If it is zero, add it to a counter *cntSame*.
3. Repeat the above process, and choose the one with highest bias at *cntSame* from 0.5.

For D_2 as the initial difference, the highest bias is $2^{-9.2}$ when the linear characteristic L_1 left rotated by 33 bits is used.

We remark that since round independence does not hold in the case of ICEPOLE, the experimental results would be better choices for the biases rather than the theoretical estimation.

From the bias above, we immediately get a distinguishing attack on ICEPOLE.

1. Generate $2^{33.9}$ pairs of two-block plaintext such that the first 1024-bit plaintext block has initial difference D_2 and the fixed values as specified above. All the other bits are random.
2. Use ICEPOLE to encrypt the plaintext blocks and then decide the 1024 bits input and output state of the first P_6 . Discard those pairs if the two bits in round 5 output in the linear characteristic L_1 left rotated 33 bits cannot be recovered. Note that for each bit, the probability is $2^{-6.45}$ to recover. So there are 2^{21} pairs left. Then compute the bit at position $S[1][1][33]$ of the output of round 5 using the recovered bits from round 6.
3. Analyze the bias of the XORed difference of those two bits ($S[1][1][33]$). If the bias is larger than $2^{-10.2}$ we conclude that it is the ICEPOLE encryption.

The success probability is computed by using the normal distribution to approximate the binomial distribution of the bias. For ICEPOLE, the bias is a random variable $X \sim N(n(1/2 + 2^{-9.2}), n(1/2 + 2^{-9.2})(1/2 - 2^{-9.2}))$, where n is the number of

pairs of the recovered 5 rounds output. When $n = 2^{21}$, the probability that $X \geq 2^{-10.2}$ is 99.3%. A random permutation, on the other hand, has its bias to be a random variable $Y \sim N(n/2, n/4)$. The value is larger than $2^{-10.5}$ with probability 0.7%. Hence, we have a very good chance to distinguish the ICEPOLE encryption from a random permutation by using $2^{35.9}$ plaintext blocks (a pair of two-block messages are counted as 4 plaintext blocks), assuming the nonce can be reused.

4.4 State Recovery Attack on ICEPOLE

In this section, we will use the differential-linear characteristics to launch state recovery attacks on ICEPOLE.

4.4.1 State recovery attacks on ICEPOLE–128 and ICEPOLE–128a

For ICEPOLE–128 and ICEPOLE–128a, there are 256 unknown bits in the state before P_6 . They are in the last column of each slice. For convenience, we denote those four 64-bit unknown words in the last column as $\{U_0, U_1, U_2, U_3\}$ according to the row index. We will recover them step by step.

4.4.1.1 Recovering U_0 and U_3

To recover the unknown bits in the state, we will analyze the input values at the active S-boxes. For D_2 , there are two active S-boxes in round 1. One has input difference 2 and the other has input difference 4. We will focus on the one with input difference 4 and denote the five input bits as $b_0, b_1, \dots, b_4$ according to their positions (b_0 being the least significant bit).

When the two bits of an ICEPOLE S-box are fixed with values $b_0 = 1$ and $b_2 = 0$, there are 8 possible values for the remaining three bits. When the input difference is 4 (at b_2), the output difference have weight 1 only when $b_1 = 1$ and $b_3 = 0$. Intuitively, the lower the weight of the output difference after the first round, the

values	bias (log based 2)
$b_1 = 0, b_3 = 0$	-13.0
$b_1 = 1, b_3 = 0$	-7.3
$b_1 = 0, b_3 = 1$	-13.9
$b_1 = 1, b_3 = 1$	-11.9

higher the probability that there is no XORed difference at the output linear mask after 5 rounds. Hence, it is possible to relate the input value of the active S-box in round 1 to the bias of XORed output difference in round 5.

We experimentally find the following biases for different values of the input bit b_1 and b_3 . Note that we collected 2^{30} data in the experiments to compute the bias and repeated for several time. When the bias is less than 2^{-14} , the experimental results were not very stable, and the average number is listed in the table. In fact, it is not necessary to consider those low biases as only the highest ones could be useful for our analysis.

Note that b_1 is related to an unknown bit in U_3 , and b_3 is related to two unknown bits, in U_0 and U_3 . So we have following relations:

$$b_1 = U_3^{31} \oplus a_0$$

and

$$b_3 = U_0^{49} \oplus U_3^{49} \oplus a_1,$$

where U^x is the x -th bit in the 64-bit word U ; a_0 and a_1 are constants which can be computed from the 1024-known bits.

We describe the state recovery process given as below:

1. Generate $2^{33.9}$ pairs of two-block plaintext satisfied following requirements. The first block of the plaintext has difference D_2 and each active S-box has fixed values ‘1’ and ‘0’ in bit 0 and 2 respectively. All the other bits are random.

2. Use ICEPOLE to encrypt the plaintext blocks and then decide the 1024 bits input and output state of the first P_6 . Discard the pairs if the two bits in round 5 output in the linear characteristic L_1 left rotated 33 bits cannot be recovered. There are 2^{21} pairs left. Then compute the bit at position $S[1][1][33]$ of the output of round 5 using the recovered bits from round 6.
3. If the two bits at the position $S[1][1][33]$ are the same, we compute the values of a_0 and a_1 according to the input and increase the counter for the value of (a_0, a_1) by 1.
4. Suppose the largest counter of (a_0, a_1) takes value $a_0 = v_0$ and $a_1 = v_1$, we guess that $U_3^{31} = v_0 \oplus 1$ and $U_0^{49} \oplus U_3^{49} = v_1$.
5. By rotating the differential-linear characteristic for the other 63 bits, we can recover the two 64-bit unknown words U_0 and U_3 .

The success probability of this scheme is equivalent to the probability that a random variable $X \sim N(n(1/2 + 2^{-7.3}), n(1/2 + 2^{-7.3})(1/2 - 2^{-7.3}))$ has value greater than the random variable $Y \sim N(n(1/2 + 2^{-11.9}), n(1/2 + 2^{-11.9})(1/2 - 2^{-11.9}))$. When $n = 2^{19}$, the probability is almost 1. Since we have 2^{21} pairs of input, each of the four choices of b_0 and b_1 will be around 2^{19} pairs.

We remark that here the probability is high enough such that even if the experiment is repeated for 64 times, the success probability is still close to 1.

4.4.1.2 Recovering U_2

Assuming that U_0 and U_3 have been recovered correctly, we can use similar method to recover U_2 . In this case, we use D_1 and L_2 left rotated by 58 bits as the differential-linear characteristic.

The difference of state before the first round S-box for D_1 is given in the Table 4.6.

Table 4.6: The difference of state before the first S-box in D_1 .

0x0	0x0	0x0	0x0	0x0
0x0	0x4	0x0	0x0	0x0
0x0	0x200000	0x0	0x0	0x0
0x0	0x0	0x0	0x0	0x0

Fixed bits: With the knowledge of U_0 and U_3 , we fix bit 0, 2, 3 with the values (1, 0, 0) respectively for the active S-box at the second row. This is to ensure the output difference of this active S-box has weight exactly 1. And we fix bit 0, 4 with the values (1, 1) for the active S-box at the third row. Then, the weight of output difference can be distinguished from the input value of bit 2, which is denoted as b_2 .

We experimentally find the following biases for b_2 after 5 rounds. The biases are based on the difference of the output bit at position $S[3][1][58]$.

values	bias (log based 2)
$b_2 = 0$	-11.0
$b_2 = 1$	-15.4

From the active S-box at the third round, it is possible to find following relation:

$$b_2 = U_2^{24} \oplus a,$$

where the constant a can be computed from the known input bits.

The state recovery process process:

1. Generate $2^{36.7}$ pairs of two-block plaintext satisfying following requirements.
The first block of the plaintext has difference D_1 and each active S-box has fixed values according to the above paragraph. All the other bits are random.
2. Use ICEPOLE to encrypt the plaintext blocks and then decide the 1024 bits input and output state of the first P_6 . Discard the pairs if the two bits in the

Table 4.7: The differential of state before the first S-box in D_3 .

0x0	0x0	0x8000000000000000	0x0	0x0
0x8000	0x0	0x0	0x0	0x0
0x0	0x0	0x0	0x0	0x0
0x0	0x0	0x0	0x0	0x0

linear relation (according to L_2 rotated by 58 bits) in the output of round 5 cannot be recovered. There are 2^{25} pairs left. Then compute the bit at position $S[3][1][58]$ of the output of round 5 using the recovered bits from round 6.

3. If the two bits at the position $S[3][1][58]$ are the same, we compute the value of a according to the input and increase the counter for that value by 1.
4. Suppose the largest counter of a take value $a = v$, we guess that $U_1^0 = v$.
5. By rotating the differential-linear characteristic for the other 63 bits, we can recover the 64-bit unknown word U_1 .

The estimated success probability is 99.6% for each bit.

4.4.1.3 Recovering U_1

At this stage, we assume that U_0 , U_2 and U_3 have been recovered correctly. In this case, we select D_3 and L_2 left rotated by 35 bits as the differential-linear characteristic.

The differential of state before the first round S-box for D_3 is given in the Table 4.7.

Fixed bits: We fix bit 1, 2, 4 with the values (1, 0, 1) for the active S-box at the first row. It is to ensure that the weight of output difference of this active S-box is determined by the value of input bit 3 which is denoted as $b_{0,3}$. And we fix bit 0, 2, 3, 4 with the values (0, 1, 1, 1) for the active S-box at the second row. It is to ensure that the weight of output difference is determined by the value of input bit 1 which is denoted as $b_{1,1}$.

values	bias (log based 2)
$b_{0,3} = 0, b_{1,1} = 0$	-11.2
$b_{0,3} = 1, b_{1,1} = 0$	-15.2
$b_{0,3} = 0, b_{1,1} = 1$	-16.4
$b_{0,3} = 1, b_{1,1} = 1$	-14.8

The $b_{0,3}$ and $b_{1,1}$ are related to the unknown bits:

$$b_{0,3} = a_0 \oplus U_1^{12}$$

$$b_{1,1} = a_1 \oplus U_1^{13}$$

where a_0 and a_1 are constants which can be computed from the known input.

We experimentally find the following biases for different values of the input bit $b_{0,3}$ and $b_{1,1}$. The biases are base on the difference of the output bit at position $S[3][1][35]$.

We remark that the biases other than the first row in above table may not be very accurate considering the small bias.

The state recovery process process:

1. Generate $2^{37.7}$ pairs of two-block plaintext satisfied following requirements. The first block of the plaintext has difference D_3 and each active S-box has fixed values according to the above paragraph. All the other bits are random.
2. Use ICEPOLE to encrypt the plaintext blocks and then decide the 1024 bits input and output state of the first P_6 . Discard those pairs if the two bits in the linear relation (according to L_2 rotated by 35 bits) in the output of round 5 cannot be recovered. There are 2^{26} pairs left. Then compute the bit at position $S[3][1][35]$ of the output of round 5 using the recovered bits from round 6.
3. If the two bits at the position $S[3][1][35]$ are the same, we compute the value of a_0 and a_1 according to the input and increase the counter for (a_0, a_1) by 1.

4. Suppose the largest counter of (a_0, a_1) take value $a_0 = v_0$ and $a_1 = v_1$, we guess that $U_1^{12} = v_0$ and $U_1^{13} = v_1$.
5. By rotating the differential-linear characteristic for the other 31 even bits less than 64, we can recover the 64-bit unknown word U_1 .

Note that in each rotation of the differential-linear characteristic, we are able to recover two consecutive bits, so we only need to test 32 rotations to recover the 64-bit U_1 . The estimated success probability is 98.7% for each bit.

4.4.1.4 Correcting the recovered state

In the previous state recovery attack, only the first 128 bits can be correctly recovered with probability almost 1. For the other 128 bits, although the success probability is around 99%, it is still possible that the state we recovered has some error bits. Whether the state is correct can be easily verified by encrypting some new messages and compare the ciphertext.

We can correct up to 7 error bits with relatively low complexity. To correct i error bits is to choose any i bits from the 128 bits and flip the values. Then test whether the modified unknown state is correct.

Suppose that for the 128 bits U_1 and U_2 , the probability that each bit is correct is 0.99, we can compute the probability that the number of error bits is less than 8 as

$$\sum_{i=0}^7 \binom{128}{i} \times .99^{128-i} \times .01^i = 0.99995.$$

The total number of encryptions to correct up to 7 error bits is $2^{37.5}$, which is negligible to the whole attack.

4.4.1.5 Summary of the attack

The data complexity is:

- U_0 and U_3 : $2 \times 2 \times 2^{33.9} \times 2^6$. We multiply $2^{33.9}$ by 2 two times due to the fact that we use $2^{33.9}$ pairs of 2-blocks plaintext.
- U_2 : $2 \times 2 \times 2^{36.7} \times 2^6$
- U_1 : $2 \times 2 \times 2^{37.7} \times 2^5$
- Total: $2^{45.8}$

The time complexity is the $2^{45.8}$ encryptions of the one block plaintext and the possible $2^{37.5}$ encryptions for correction.

The memory cost is mainly on the storage of some counters, which is negligible.

The success rate of this attack is close to 1, and can be adjusted through the number of input messages.

4.4.2 State recovery attack on ICEPOLE–256a

In the case of ICEPOLE–256a, there are 320 unknown bits in the input and output states in the encryption of a block. In addition to the U_0 to U_3 in the previous subsection, we use U_4 to denote the unknown 64-bit word $S[3][3]$.

Different from the ICEPOLE–128 and ICEPOLE–128a, in the last row of the output in ICEPOLE–256a, there are only 3 known bits instead of 4 known bits. Consequently, it is impossible to recover the input bits given the output bits for that row. To deal with this issue, we have to consider the linear relation of the input mask of the S-box. When the value of the input mask is less than 8, the largest bias is $3/16$, but if the value of the input mask is 8, the largest bias is $1/16$.

For L_1 , there are two mask bits placed in the last row, one is 4 and the other is 8. From the Piling-Up Lemma [96], the bias is $2^{-5.4}$.

For L_2 , there are three mask bits placed in the last row, 2, 4, and 8. By Piling up Lemma, the bias is $2^{-6.8}$.

To increase the bias of the last row, we introduce another linear characteristic L_3 (Table 4.4.2), which has only 1 bit in the last row of the input mask of round 6 S-box.

Table 4.8: The linear characteristic 3, assuming S-boxes are identical mappings
(a) input linear mask

0x0	0x10000000000000	0x20000	0x820000000000	0x40000000000000
0x41000000000000	0x0	0x800000000000	0x20000	0x2000000000
0x40000000000000	0x10000000000000	0x2000000000	0x80000020000	0x0
0x41000000000000	0x0	0x10000000000000	0x2000020000	0x80000000000

(b) output linear mask

0x40000	0x0	0x0	0x800000000000	0x200000000000
0x8000	0x4	0x0	0x0	0x0
0x8	0x200000	0x0	0x0	0x100000000000
0x0	0x0	0x200000000000	0x0	0x0

values	bias (log based 2)
$b_{3,2} = 0$	-9.2
$b_{3,2} = 1$	-15.3 (negative)

For L_3 , the bias is $3/16$ and the probability to recover other bits is $2^{-9.6}$.

To recover the U_4 , we use the differential characteristic D_2 with linear characteristic L_1 left rotated 33 bits, same as the recovery of U_0 and U_3 . Since the value of $S[3][2]$ is not known, we will only fix the active bits in $S[3][0]$, setting it to 1.

We will find the value of bit 2 in the input of active S-box related to $S[3][1]$ with difference $0x400$. We denote this bit $b_{3,2}$, and we have $b_{3,2} = a \oplus U_4^{33}$, where a is a constant from the known input.

We experimentally find the following biases for the input bit values $b_{3,2}$ after 5 rounds. The biases are base on the difference of the output bit at position $S[1][1][33]$.

Considering the linear relation of this XORed value to the 4 bits in the two output states, the bias of the XORed difference of the 4 bits becomes 2^{-18} by Piling up lemma. To ensure the high probability of correctly guessing the value, we need around 2^{40} pairs of two-block plaintext. The total data needed is around $2^{58.7}$.

Next, the complexity of recovering the other unknown words need to be amended. We omit the similar attack process and discuss the estimated complexities here.

For U_0 and U_3 , the increased bias is $2^{-8.8}$ which needs to be compensated by around $2^{17.6}$ additional messages. Furthermore, some of the bits from the S-box layer

in round 6 can directly be recovered from the linear relation, hence increasing the success probability by a factor of 2^4 . The total effect is the data complexity becomes $2^{55.5}$.

For U_2 , D_1 with L_3 left rotated 13 bits will be used and for U_1 , D_3 with L_1 left rotated 2 bits will be used. The increased bias is roughly $2^{-2.8}$ for both cases (including the increased 5-round bias). And the decreased probability for recovering the values before round 6 S-box has a factor $2^{-7.5}$. Therefore, the total data complexity for recovering each bit is increased by $2^{13.1}$, which is $2^{57.8}$.

Therefore, the total data complexity of the state recovery attack for ICEPOLE-256a is estimated to be $2^{59.8}$, which is less than the constraint 2^{62} .

4.4.3 Implications of the state recovery attacks

For ICEPOLE-128, the state recovery attack implies the failure of encryption security if both the nonce and the secret message number are reused. When an adversary has the full knowledge of a state, he can invert the cipher until the secret message number is injected. Thus, the adversary can decrypt arbitrary plaintext blocks. It also implies a forgery attack on the authentication of plaintext and associated data since the valid tag for any modified associated data or plaintext block can be computed. Since both the key and secret message number are unknown, the adversary cannot recover the key.

For ICEPOLE-128a and ICEPOLE-256a, the state recovery attack implies the whole security is broken if the nonce is reused. The initialization of ICEPOLE-128a is invertible, so the adversary can directly recover the secret key from the known state. Then both the encryption and authentication are insecure.

Summary of the security of ICEPOLE under our analysis when nonce is reused is given in Table [4.9](#).

Table 4.9: Number of bits of security when nonce and secret message number can be reused.

	ICEPOLE-128	ICEPOLE-128a	ICEPOLE-256a
confidentiality for the plaintext	46	46	60
confidentiality for the secret message number	128	-	-
integrity for the plaintext	46	46	60
integrity for the associated data	46	46	60
integrity for the secret message number	46	-	-
integrity for the public message number	46	46	60

4.5 Experimental Results

We experimentally verified the state recovery attack on ICEPOLE-128a. And we managed to recover the 256 unknown bits practically. The experiments used the state after the initialization with all zeros IV and key. The unknowns states are:

$$U_0 = 0x1e7aed5bfaeb535f$$

$$U_1 = 0xe0dcc6422595e5ba$$

$$U_2 = 0x892bf76586876c23$$

$$U_3 = 0x8b2ef3bf50e902f6$$

To recover U_0 and U_3 , we run the attack in Sect. 4.4.1.1 on a server with 64 cores (AMD Opteron(tm) Processor 6276). Instead of checking the constants from input, we used an equivalent method: directly extract the input bits of the active S-box in the first round, and decide whether they are the estimated ones. The number of plaintext pairs we used is 2^{34} for each bit. The attacks takes 15.3 hours and all the 128 bits recovered are correct.

To recover U_2 , we run the attack in Sect. 4.4.1.2 on the server. The number of plaintext pairs we used is 2^{37} for each bit. The attacks takes 3.5 days and all the bits are correct.

To recover U_1 , we run the attack in Sect. 4.4.1.3 on the server. The number of plaintext pairs we used is 2^{38} for each bit. The attacks takes 3.5 days and there is one error bit.

Since the number of error bits is very small, the experiments show that our state recovery attack indeed works for ICEPOLE-128a.

4.6 Conclusion

We analyzed the security of the ICEPOLE family of authenticated ciphers using the differential-linear cryptanalysis when nonce is misused. ICEPOLE is strong against differential cryptanalysis since only part of the input difference is affected by message in the attack (so the best differential attack against the permutation cannot be applied to break ICEPOLE); and ICEPOLE is strong against linear cryptanalysis since only part of the input and output of the permutation are known in the attack (so the best linear attack against the permutation cannot be applied to break ICEPOLE). We successfully developed the differential-linear cryptanalysis against ICEPOLE by bypassing the input/output constraints of ICEPOLE. Our attacks show that the states of all the ICEPOLE variants can be recovered, and the secret key of ICEPOLE-128a and ICEPOLE-256a can also be recovered. The security claims of ICEPOLE do not hold under the nonce misuse circumstances.

From the attacks against ICEPOLE, the lesson we learned is that when we are designing a strong cipher based on a permutation, it is better to consider the best attacks against the permutation in which the input/output can affect the whole state. Furthermore, if the performance of a cipher is improved by considering input/output constraints, the designers should analyze whether the input/output constraints could be bypassed in the attacks.

Chapter 5

Cryptanalysis of the Authenticated Encryption Algorithm COFFE

5.1 Introduction

COFFE is a hash-function-based authenticated encryption scheme designed by Forler *et al.* [67]. COFFE is designed to be secure for computationally constraint environment. As mentioned above, COFFE utilises a hash function for both encryption and authentication without introducing any block cipher primitive. According to [67], COFFE is one of the first authenticated encryption that is purely based on hash function. This alternative direction of constructing an authenticated encryption system is interesting for constructing a secure authenticated encryption.

The designers claim that COFFE is secure against chosen plaintext attack in nonce-respecting scenario. It is also claimed to have ciphertext-integrity even in nonce-misuse scenario. In particular, it is claimed that the ciphertext integrity of COFFE is at least strong as the indistinguishability of the hash function used. That is, forging a ciphertext with a valid tag should be as hard as finding collision in the underlying hash function. Furthermore, it also provides additional features. Firstly, it provides failure-friendly authenticity, that is, COFFE provides reasonable authen-

ticity in the case of weaker underlying hash function. Secondly, it also provides side channel resistance under nonce-respecting scenario.

During the analysis we consider the scheme firstly under the nonce-repeating scenario. Instead of using any specific hash function for the underlying primitive, we analyse it on the generic construction case with an ideal underlying hash function. We show that under these settings, some instances of COFFE with particular parameters are vulnerable to distinguishing attack, ciphertext forgery attack, or related key recovery attack. Thus, the security claim of COFFE for these parameters does not hold.

The attacks come from the consideration that intermediate state values of two different COFFE instantiations can be made the same while having different inputs. The vulnerability comes from the fact that having most of the parameters to be variables, different set of parameters can be chosen and combined to create the collision. The attack starts by first trying to find a specific value for the parameters where this can happen. Having found these parameters, different approaches are made to exploit this discovery to launch either distinguishing attack, forgery attack, or key recovery attack. In this chapter, we found that for the distinguishing and forgery attack, if we use the same secret key for all the instantiations, the success probability is close to 1.

The rest of this chapter is structured as follows: The generic specification of COFFE is given in Section 5.2. Section 5.3 provides some analysis and observation of COFFE. Section 5.4 introduces the distinguishing attack. Section 5.5 provides two variants of ciphertext forgery attack. We proposed a related key recovery attack on Section 5.6. Lastly, Section 5.7 concludes the chapter.

5.2 The COFFE Authenticated Cipher

The COFFE family of authenticated ciphers uses six parameters: key length, nonce length, block size, hash function input and output size, and tag length. We will briefly

Figure 5.1: General scheme of COFFE encryption and authentication (Fig. 2 of [67])

describe the specification of COFFE authenticated cipher. The full specification can be found in [67]. An overview of COFFE is provided in Figure 5.1.

5.2.1 Notations

Throughout this chapter, we will be using the following notations:

- $\mathcal{F}$: Underlying Hash function
 - γ : Input size for $\mathcal{F}$ assuming “one compression function invocation per hash function call”
 - δ : Output size for $\mathcal{F}$
- $\mathcal{L}_K$: Secret key length expressed in bits. The length of this string should be a multiple of a byte
- $\mathcal{L}_V$: Nonce length expressed in bits. The length of this string should be a multiple of a byte
- $\mathcal{L}_T$: Tag length expressed in bits. The length of this string should be a multiple of a byte, $\mathcal{L}_T \leq \delta$

- α : Message block size
- β : Last message block size, $\beta \leq \alpha \leq \delta$
- x : Bits for domain value. The string length will follow the number of bytes needed to express $\beta + 5$ in bits.
- Let v be a binary string and b be a positive integer.
 - $|v|$: The length of v in bits.
 - $|v|_b$: A b -bit binary representation of v .
 - $[v]$: The length of v in byte.
 - $[v]_b$: A b -byte binary representation of v .
 - $\mathbf{b}$: $[\beta + 5]$.
- $\mathcal{S}_1 || \mathcal{S}_2$: Concatenation of string $\mathcal{S}_1$ followed by $\mathcal{S}_2$.
- $\mathcal{S}_1 \oplus_{\ell} \mathcal{S}_2$: The ℓ - bit strings obtained by XOR-ing the ℓ least significant bits of $\mathcal{S}_1$ and $\mathcal{S}_2$.
 - $\mathcal{S}_1 =_b \mathcal{S}_2$: The last b bits of both $\mathcal{S}_1$ and $\mathcal{S}_2$ is the same.
- $\mathcal{S}_1 || 0^* || \mathcal{S}_2$: When clear the total length should be, say a bits, concatenate $\mathcal{S}_1$ with 0-bits then with $\mathcal{S}_2$ with the number of 0-bits being the difference between a and the total length of $\mathcal{S}_1$ and $\mathcal{S}_2$.
- $\mathcal{K}$: Secret key string
- $\mathcal{V}$: Nonce
- $\mathcal{L}$: The number of message blocks for the encryption
- $\mathcal{S}$: Session Key with length δ bits
- $\mathcal{H}$: Associated Data

- $\mathcal{M}[i], 1 \leq i \leq \mathcal{L}$: The i -th message block, an α bit string except for $\mathcal{M}[\mathcal{L}]$ having length β bits.
- $\mathcal{C}[0]$: The initial vector
- $\mathcal{C}[i], 1 \leq i \leq \mathcal{L}$: The i -th ciphertext block with the same length as $\mathcal{M}[i]$
- $\mathcal{T}[i], 0 \leq i \leq L$: Chaining values for the scheme each of which having length δ bits
- $\mathcal{T}$: Message Tag.

5.2.2 Associated Data Processing

The method of processing the associated data, $\mathcal{H}$, can be divided into three cases based on the length of the associated data.

- If the length of $\mathcal{H}$ is less than δ bits, it is appended by 1 followed by appropriate number of zeros to reach δ bits. This is defined as $\mathcal{T}[0]$ and a domain value x is defined to be 1.
- If the length of $\mathcal{H}$ is exactly δ bits, this is directly defined as $\mathcal{T}[0]$ while the domain value x is set to be 2.
- If the length of $\mathcal{H}$ is more than δ bits, feed $\mathcal{H}$ to $\mathcal{F}$ and the resulting hash output is used as the value of $\mathcal{T}[0]$ and x is defined as 3.

5.2.3 Initialization

There are two values that need to be computed in the initialization phase, $\mathcal{S}$ and $\mathcal{C}[0]$. Firstly, the session key, $\mathcal{S}$ which is defined based on $\mathcal{K}, \mathcal{V}, \mathcal{L}_K, \mathcal{L}_V$, and $\mathbf{b}$. The value of $\mathcal{S}$ is defined to be $\mathcal{F}(\mathcal{K}||\mathcal{V}||0^*||\mathcal{L}_K||\mathcal{L}_V||[\mathbf{0}]_b)$. Note that here 0^* is used to pad the string to make the length equals to γ .

Next, the constant $\mathcal{C}[0]$ which depends only on the message block size α . $\mathcal{C}[0]$ is defined to be the first $\frac{\alpha}{4}$ post-decimal values of π interpreted as a hexadecimal string. So for example, since the decimal values of π is $.14159\dots$, if $\alpha = 16$, Then $\mathcal{C}[0] = 0x1415 = 0001010000010101$.

5.2.4 Processing plaintext

Plaintext is encrypted to obtain the ciphertext after the generation of session key $\mathcal{S}$, the initialization vector $\mathcal{C}[0]$, initial chain value $\mathcal{T}[0]$ and the domain value, x . The plaintext blocks are processed as follow:

$$\mathcal{T}[1] = \mathcal{F}((\mathcal{S} \oplus \mathcal{T}[0]) \parallel \mathcal{C}[0] \parallel 0^* \parallel [x]_{\mathbf{b}})$$

$$\mathcal{C}[1] = \mathcal{M}[1] \oplus_{\alpha} \mathcal{T}[1]$$

for all blocks $\mathcal{M}[i], 2 \leq i \leq \mathcal{L} - 1 \{$

$$\mathcal{T}[i] = \mathcal{F}((\mathcal{S} \oplus \mathcal{T}[i-1]) \parallel \mathcal{C}[i-1] \parallel 0^* \parallel [4]_{\mathbf{b}})$$

$$\mathcal{C}[i] = \mathcal{M}[i] \oplus_{\alpha} \mathcal{T}[i]$$

$\}$

$$\mathcal{T}[\mathcal{L}] = \mathcal{F}((\mathcal{S} \oplus \mathcal{T}[\mathcal{L}-1]) \parallel \mathcal{C}[\mathcal{L}-1] \parallel 0^* \parallel [4]_{\mathbf{b}})$$

$$\mathcal{C}[\mathcal{L}] = \mathcal{M}[\mathcal{L}] \oplus_{\beta} \mathcal{T}[\mathcal{L}].$$

5.2.5 Tag generation

After the associated data and plaintext are processed, the $\mathcal{L}_T$ -bit tag $\mathcal{T}$ is derived:

$$\mathcal{T} = \mathcal{F}((\mathcal{S} \oplus \mathcal{T}[\mathcal{L}]) \parallel \mathcal{C}[\mathcal{L}] \parallel 0^* \parallel \mathcal{L}_T \parallel \beta + 5)$$

The decryption is trivial and we omit it here. For the verification, only the $\mathcal{L}_T$ least significant bits of the tags are checked.

5.2.6 Security goals of COFFE

COFFE is claimed to have the INT-CTXT (ciphertext integrity) and IND-CPA (indistinguishable under chosen plaintext attack) property under nonce-respecting scenario.

In particular, in lemma 1 of [67], we have:

Lemma 5.2.1. *[67] Let Π be a COFFE scheme as defined above with $\mathcal{F}$ as its underlying hash function. Then the advantage of adversary $\mathcal{A}$ under nonce-respecting scenario with q queries and ℓ message blocks to the encryption oracle with time bounded by t can be upper bounded by:*

$$\mathbf{Adv}_{\Pi}^{CPA}(q, \ell, t) \leq \frac{8\ell^2 + 3q^2}{2^n} + 2 \cdot \mathbf{Adv}_{\mathcal{F}}^{PRF-XRK}(q, \ell, t).$$

In other words, distinguishing COFFE from a random function with chosen input under the bound of (q, ℓ, t) should be at least as hard as distinguishing $\mathcal{F}$ from a random function $\$: \{0, 1\}^\gamma \Rightarrow \{0, 1\}^\delta$.

Additionally, COFFE has some other security claim under different circumstances. Firstly, under the nonce-misuse scenario, it claimed that

- “..., the integrity of the ciphertext does not depend on a nonce, but only on the security of $\mathcal{F}$ ”.

In particular, in lemma 2 of [67], we have:

Lemma 5.2.2. *[67] Let Π be a COFFE scheme as defined above with $\mathcal{F}$ as its underlying hash function. Then in the nonce-ignoring adversary scenario with q queries for ℓ message blocks and t times, we have*

$$\mathbf{Adv}_{\Pi}^{INT-CTXT}(q, \ell, t) \leq \frac{3\ell^2 + 2q^2}{2^\delta} + \frac{q}{2^{\mathcal{L}_T}} + \mathbf{Adv}_{\mathcal{F}}^{PRF}(q + \ell, O(t)).$$

This implies that the hardness of forging a ciphertext with a valid tag should be at least as hard as distinguishing $\mathcal{F}$ from a random function from $\{0, 1\}^\gamma$ to $\{0, 1\}^\delta$.

Secondly, COFFE also provides a failure-friendly authenticity. That is, under a weaker assumption on the security of the underlying hash function $\mathcal{F}$, the authenticity of the message is still kept.

Lastly, COFFE also provides a reasonable resistance against side channel attack. This is so because “for each encryption process, a new short term key is derived from a nonce and the long term key” [67].

5.3 Analysis on the COFFE scheme

In our analysis, we assume $\mathcal{F} : \{0, 1\}^* \Rightarrow \{0, 1\}^\delta$ to be an arbitrary ideal hash function with γ being the largest possible length of the input to ensure exactly one compression function invocation per hash function call. Here we are assuming the possibility of the parameters to have length more than 255 bits. In other words, it is possible that it requires more than 1 byte to represent $\mathcal{L}_K, \mathcal{L}_V, \mathcal{L}_T$ in their binary format.

The first observation is about the input for the hash function call. Note that since we are only considering concatenation, there is not always a way, given the concatenated string, to uniquely determine the value for each strings before the concatenation. For example, if $a||b = 11011$, it is possible for $a = 110, b = 11$ or $a = 1, b = 1011$. This leads to the possibility that two different sets of strings to be concatenated to the same string.

Observation 5.3.1. *For the input of any hash function call, due to the absence of separator between substrings and changeable elements lengths, it is possible to have two different sets of strings to be concatenated to the same string.*

On the following subsections, we analyse this observation further to find whether it is possible to utilise this to cause a collision in the intermediate state value of the COFFE. We first consider the case when we fix the message block size while allowing two different last message block sizes, β_1 and β_2 , to be used. The analysis is focused on the case when $|\beta_1 - \beta_2|$ is a multiple of 256. The analysis on this can be

found on Subsection 5.3.1. Next we also consider the possibility of having identical intermediate state values when we change the message block size, α , while keeping β fixed. The analysis is focused on how α should be chosen in such a way for the first message block encryption of both instantiations to have identical hash value output. This is discussed in Subsection 5.3.2.

5.3.1 Modification of β

We fix α and consider different values of β . In our next observation, with large enough α , it is possible to have $\beta_1 < \beta_2 \leq \alpha$ such that $\beta_2 - \beta_1$ is a multiple of 256. This implies that the last byte of the input of $\mathcal{F}$ in the tag generation for the two different plaintexts can be the same. As discussed above, however, we want the collision to happen in the whole input string for any $\mathcal{F}$ input. If both $\beta_1 + 5$ and $\beta_2 + 5$ require 2 bytes to represent in binary format, the second to last byte will never agree. So for collision to happen, we need $\beta_1 + 5 < 256$, $256 \leq \beta_2 + 5 < 65536$ and $\beta_2 = \beta_1 + 256\rho$ for some integer $1 \leq \rho \leq 255$.

To further analyse this observation, we consider the note by the designers regarding the increase of number of byte required for the binary representations of the domain. In [67], it is stated that if $\beta + 5$ exceeds one byte, all domain representations in the current COFFE will be encoded as two-byte values instead of one. So this is important in our analysis on the possibility of exploring this observation to introduce a successful attack.

Note that in the message processing, assuming that γ is big enough, there are enough bits of the zero padding between $\mathcal{C}[i]$ and the domain values for the encryption to absorb the additional byte for the domain values in case $\beta + 5$ is increased from one byte to two bytes value. So the parts that need to be taken care of for this to happen are the session key generation and tag generation.

In the session key generation, we consider the last several bytes of the input of $\mathcal{F}$. Here we have the input to be $\dots \parallel a \parallel \mathcal{L}_K \parallel \mathcal{L}_V \parallel 0$. Note that when we expand the

domain value from 1- to 2-byte value, the domain value should still have the same value. So the second to last byte for the input must be 0. This gives us our next observation.

Observation 5.3.2. *To ensure that collision can occur when extending the domain from 1- to 2-byte value, the initial value of $\mathcal{L}_V$ must be a multiple of 256. This means that if the initial $\mathcal{L}_V$ is a 1-byte value, it must be 0, that is, no nonce in the first instance.*

Our primary goal in this section is to investigate the possibilities to have two different (key, nonce) pairs, $(\mathcal{K}_1, \mathcal{V}_1)$ and $(\mathcal{K}_2, \mathcal{V}_2)$ with lengths $\mathcal{L}_{K_1}, \mathcal{L}_{V_1}, \mathcal{L}_{K_2}, \mathcal{L}_{V_2}$ respectively such that

$$(\mathcal{K}_1 \parallel \mathcal{V}_1 \parallel 0^* \parallel \mathcal{L}_{K_1} \parallel \mathcal{L}_{V_1} \parallel [0]_1) = (\mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^* \parallel \mathcal{L}_{K_2} \parallel \mathcal{L}_{V_2} \parallel [0]_2).$$

For simplicity, let

$$\mathcal{S}_1 = (\mathcal{K}_1 \parallel \mathcal{V}_1 \parallel 0^* \parallel \mathcal{L}_{K_1} \parallel \mathcal{L}_{V_1} \parallel [0]_1),$$

$$\mathcal{S}_2 = (\mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^* \parallel \mathcal{L}_{K_2} \parallel \mathcal{L}_{V_2} \parallel [0]_2).$$

Here, we note that collision is indeed possible as illustrated by the following example: Let SHA-512 be our hash function. We can choose any 256-bit string, say $\mathcal{K}$, and set $\mathcal{K}_1 = \mathcal{K}_2 = \mathcal{K}$. Now we use any one bit value (0 or 1) as our $\mathcal{V}_2$ while $\mathcal{V}_1$ is set to be $\mathcal{V}_2$ appended by 255 zeros. Now if we use 472 zero paddings on the first string while using 727 bits for the second string we will have

$$\mathcal{S}_1 = \mathcal{S}_2 = (\mathcal{K} \parallel 1 \parallel 0^{255} \parallel 0^{472} \parallel [1]_1 \parallel [0]_1 \parallel [1]_1 \parallel [0]_1 \parallel [0]_1).$$

In the remaining of this section, we will try to analyse whether such collision is possible for other instances of COFFE. Here we analyse different cases of $\mathcal{S}_1$ on the possibility of having $\mathcal{S}_1 = \mathcal{S}_2$. The factors that we need to consider are the number of bytes required for $\mathcal{L}_{K_i}, \mathcal{L}_{V_i}$ and whether there is any zero paddings required. Note

that if $\mathcal{L}_{K_i}$ or $\mathcal{L}_{V_i}$ is a 3-byte value, the value will be at least 65536 which is too big. To simplify our discussion, for this chapter, we will only consider the key and nonce to have length whose binary format can be represented as at most a 2-byte value. Due to the big number of cases we need to consider and the similarity of the cases, we will just discuss one case as example and a full analysis of the other cases can be found in Appendix A.1 while the Table 5.1 containing the conclusion is provided for reference.

I.3.c Case I.3.c.: $\mathcal{S}_1$ has no zero paddings, $[\mathcal{L}_{K_1}] = 1, [\mathcal{L}_{V_1}] = 2, [\mathcal{L}_{K_2}] = 1, [\mathcal{L}_{V_2}] = 2$.

By Observation 5.3.2, $\mathcal{L}_{V_1} = 256\mathbf{b}$ and $\mathcal{L}_{K_1} = \mathbf{a}$ where $1 \leq \mathbf{a}, \mathbf{b} \leq 255$. Both $\mathbf{a}$ and $\mathbf{b}$ are nonzero because of the following reasons. First of all, since $\mathcal{L}_{K_1} = \mathbf{a}$, if $\mathbf{a} = 0$, then there is no secret key, in which case, no confidentiality for the message. So we can disregard the case when $\mathcal{L}_K = 0$. Next, since $[\mathcal{L}_{V_1}] = 2$, this should mean that $\mathcal{L}_{V_1} \geq 256$ since otherwise, $[\mathcal{L}_{V_1}] = 1$. So if $\mathbf{b} = 0$, this implies $\mathcal{L}_{V_1} = 0$ which violates the requirement $\mathcal{L}_{V_1} \geq 256$. Hence

$$\mathcal{S}_1 = (\mathcal{K}_1 \parallel \mathcal{V}_1 \parallel \mathbf{a} \parallel \mathbf{b} \parallel [0]_2).$$

Let $\mathcal{V}_1 = \mathcal{V}'_1 \parallel \mathbf{d}$ where $\mathbf{d}$ is the last byte of $\mathcal{V}_1$. So

$$\mathcal{S}_1 = (\mathcal{K}_1 \parallel \mathcal{V}'_1 \parallel \mathbf{d} \parallel \mathbf{a} \parallel \mathbf{b} \parallel [0]_2).$$

Consider the alternative string $\mathcal{S}_2$. Recall that here we want $\mathcal{S}_1 = \mathcal{S}_2$ where $\mathcal{S}_2$ has its domain value represented as a 2-bytes value. This implies that the $[0]_2$ in the last 2 bytes of $\mathcal{S}_1$ must appear as the domain for $\mathcal{S}_2$. So this implies that $\mathcal{L}_{K_2} = \mathbf{d}$ and $\mathcal{L}_{V_2} = 256\mathbf{a} + \mathbf{b}$. Let t' be the number of zero padding in $\mathcal{S}_2$ where $t' \geq 0$. Equating $\mathcal{S}_1$ with $\mathcal{S}_2$, we have $\mathcal{K}_1 \parallel \mathcal{V}'_1 = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'}$. Now comparing the length of these substrings, we have $\mathbf{a} + 256\mathbf{b} - 8 = \mathbf{d} + 256\mathbf{a} + \mathbf{b} + t'$ or equivalently, $255(\mathbf{b} - \mathbf{a}) = \mathbf{d} + 8 + t'$. Consider the family:

$$\mathcal{F}_{(1,2),(1,2)} = \{(\mathbf{a}, \mathbf{b}, \mathbf{d}, t') : 1 \leq \mathbf{a}, \mathbf{b}, \mathbf{d} \leq 255, t' \geq 0, 255(\mathbf{b} - \mathbf{a}) = \mathbf{d} + 8 + t'\}.$$

Now we consider the feasibility of each element of $\mathcal{F}_{(1,2),(1,2)}$. Feasibility here means the possibilities of using these values as the parameters to have the collision. Let $(\mathbf{a}, \mathbf{b}, \mathbf{d}, t') \in \mathcal{F}_{(1,2),(1,2)}$. Note that the collision may not happen with probability 1 due to the case when $\mathcal{K}_1 \neq \mathcal{K}_2$. Note that since key is the first part of the collided string, this can only happen when $\mathcal{L}_{K_1} \neq \mathcal{L}_{K_2}$.

Before going on to the analysis, we have an assumption first. Suppose that $\mathcal{L}_{K_1} > \mathcal{L}_{K_2}$. Since $\mathcal{K}_1$ is the first $\mathcal{L}_{K_1}$ bits of $\mathcal{S}_1$, $\mathcal{K}_2$ is the first $\mathcal{L}_{K_2}$ bits of $\mathcal{S}_2$ and we need $\mathcal{S}_1 = \mathcal{S}_2$, the first $\mathcal{L}_{K_2}$ bits of $\mathcal{K}_1$ must be $\mathcal{K}_2$. Instead of assuming that this happens by chance, we will assume the following: The user has 2 different instantiations of COFFE scheme with different parameter and different key length. However, the keys chosen by the user are not independent. The longer key is an extension of the shorter key by a random secret string. We note that this assumption is only made for the related key setting attack and not for the general attack.

Based on this assumption, we then have the probability of $\mathcal{K}_1$ to have its first $\mathcal{L}_{K_2}$ bits to be the same as $\mathcal{K}_2$ is exactly 1.

Now back to our case, we have that $\mathcal{L}_{K_1} = \mathbf{a}$ and $\mathcal{L}_{K_2} = \mathbf{d}$. So the length difference of the two keys is $|\mathbf{a} - \mathbf{d}|$ bits. Now if $\mathbf{a} = \mathbf{d}$, then we have $\mathcal{K}_1 = \mathcal{K}_2$. Now the rest of the two strings are $\mathcal{V}'_1 \parallel \mathbf{d} \parallel \mathbf{a} \parallel \mathbf{b} \parallel [0]_2$ and $\mathcal{V}_2 \parallel 0^{t'} \parallel \mathbf{d} \parallel \mathbf{a} \parallel \mathbf{b} \parallel [0]_2$. So we have $\mathcal{V}_1 = \mathcal{V}_2 \parallel 0^{t'} \parallel \mathbf{d}$. Now since $\mathcal{V}_1$ can be controlled by the attacker, we can easily set this to be true. So the probability of the two strings to collide is 1 if $\mathbf{a} = \mathbf{d}$.

Now consider when $\mathbf{a} \neq \mathbf{d}$, specifically, $\mathbf{a} > \mathbf{d}$. The other case can be analysed using exactly the same way. Now let $\mathcal{K}_1 = \mathcal{K}_2 \parallel \mathcal{K}'_1$ where $\mathcal{K}'_1$ is the last $\mathbf{a} - \mathbf{d}$ bits of $\mathcal{K}_1$. We have $\mathcal{K}'_1 \parallel \mathcal{V}'_1 \parallel \mathbf{d} \parallel \mathbf{a} \parallel \mathbf{b} \parallel [0]_2$ and $\mathcal{V}_2 \parallel 0^{t'} \parallel \mathbf{d} \parallel \mathbf{a} \parallel \mathbf{b} \parallel [0]_2$ as the remaining part of the two strings truncating the first $\mathbf{d}$ bits. Thus, $\mathcal{K}'_1 \parallel \mathcal{V}'_1 = \mathcal{V}_2 \parallel 0^{t'}$. Note that since $1 \leq \mathbf{a}, \mathbf{d} \leq 255$, $\mathbf{a} - \mathbf{d} \leq 255$, $\mathcal{V}_2$ has length

$256\mathbf{a} + \mathbf{b} \geq 256$. So the entire $\mathcal{K}'_1$ is in $\mathcal{V}_2$. In other words, for the two strings to collide, we need the last $\mathbf{a} - \mathbf{d}$ bits of $\mathcal{V}_2$ must be equal to $\mathcal{K}'_1$. Since $\mathcal{K}'_1$ is supposed to be unknown, the probability of this collision is $2^{\mathbf{a}-\mathbf{d}}$. It is easy to see that the remaining substring can be set to collide with probability 1. So the probability of collision to happen is $2^{-(\mathbf{a}-\mathbf{d})}$. Using exactly the same analysis, we will see that when $\mathbf{d} > \mathbf{a}$, the probability of collision to happen is $2^{-(\mathbf{d}-\mathbf{a})}$. Hence, for any non-negative integer k , we can define a subfamily of $\mathcal{F}_{(1,2),(1,2)}$,

$$\mathcal{F}_{(1,2),(1,2),k} = \{(\mathbf{a}, \mathbf{b}, \mathbf{d}, t') \in \mathcal{F}_{(1,2),(1,2)} : |\mathbf{a} - \mathbf{d}| \leq k\}.$$

Then for any quadruplet $(\mathbf{a}, \mathbf{b}, \mathbf{d}, t') \in \mathcal{F}_{(1,2),(1,2),k}$ we take as parameter, the collision probability is at least 2^{-k} .

We remark that this probability is applicable for any choices of $\mathcal{K}_1$ and $\mathcal{K}_2$. This observation is essential in our attacks later to decide whether the attacks are only applicable to a family of key to any value of key with the given length.

We include in Table 5.1 the full list of conclusion of the session key generation analysis. Here we use t to represent the zero padding for $\mathcal{S}_1$ and t' for $\mathcal{S}_2$. In the Collision column, No means a collision in this case is impossible, yes means a collision will always happen on any choices of the parameter values (with appropriate choice of key and nonce). Lastly, $\mathcal{F}_{(a,b),(c,d)}$ or $\mathcal{F}_{\mathbf{p},(a,b),(c,d)}$ is the family of parameter values that belongs to the respected case that collision is possible. The definition of each of the family can be found in the complete analysis of each case that is either can be found above or in Appendix A.1.

Table 5.1: Session Key Generation Input Collision

$[\mathcal{L}_{\mathbf{K}_1}]$	$[\mathcal{L}_{\mathbf{V}_1}]$	$\mathbf{t}$	$[\mathcal{L}_{\mathbf{K}_2}]$	$[\mathcal{L}_{\mathbf{V}_2}]$	$\mathbf{t}'$	Collision?	Probability	Key restriction
1	1	Any	Any	Any	Any	No	0	Not Applicable
2	1	Any	1	1	Any	No	≈ 0	Not Applicable
2	1	0	2	1	Any	$\mathcal{F}_{(2,1),(2,1)}$	$2^{-(\mathcal{L}_{K_1}-\mathcal{L}_{K_2})}$	$\mathcal{K}_1 =_{(t'+8)} 0^{t'} \parallel \lfloor \frac{\mathcal{L}_{K_2}}{256} \rfloor$
2	1	$0 < t < 8$	2	1	Any	$\mathcal{F}_{\mathbf{p},(2,1),(2,1)}$	$2^{-(\mathcal{L}_{K_1}-\mathcal{L}_{K_2})}$	$\mathcal{K}_1 =_{(t'+8-t)} 0^{t'} \parallel \lfloor \frac{\mathcal{L}_{K_2}}{2^{8+t}} \rfloor$
2	1	Any	Any	2	Any	No	0	Not Applicable
1	2	Any	1	1	$255\mathcal{L}_{V_2} + t$	Yes	1	No
1	2	Any	2	1	Any	No	≈ 0	Not Applicable
1	2	0	1	2	Any	$\mathcal{F}_{(1,2),(1,2)}$	$2^{- \mathcal{L}_{K_1}-\mathcal{L}_{K_2} }$	No
1	2	$0 < t < 8$	1	2	Any	$\mathcal{F}_{\mathbf{p},(1,2),(1,2)}$	$2^{- \mathcal{L}_{K_1}-\mathcal{L}_{K_2} }$	No
1	2	0	2	2	Any	$\mathcal{F}_{(1,2),(2,2)}$	$2^{-(\mathcal{L}_{K_2}-\mathcal{L}_{K_1})}$	No
1	2	$0 < t < 16$	2	2	Any	$\mathcal{F}_{\mathbf{p},(1,2),(2,2)}$	$2^{-(\mathcal{L}_{K_2}-\mathcal{L}_{K_1})}$	No
2	2	Any	1	1	Any	No	0	Not Applicable
2	2	Any	2	1	$255\mathcal{L}_{V_2} + t$	Yes	1	No
2	2	Any	1	2	Any	No	≈ 0	Not Applicable
2	2	0	2	2	Any	$\mathcal{F}_{(2,2),(2,2)}$	$2^{- \mathcal{L}_{K_1}-\mathcal{L}_{K_2} }$	$\mathcal{K}_1 =_{\max(0,t'-(\mathcal{L}_{V_1}-8))} 0$
2	2	$0 < t < 8$	2	2	Any	$\mathcal{F}_{\mathbf{p},(2,2),(2,2)}$	$2^{- \mathcal{L}_{K_1}-\mathcal{L}_{K_2} }$	$\mathcal{K}_1 =_{\max(0,t'-(\mathcal{L}_{V_1}-(8-t)))} 0$

Recall that in the tag generation, given $\mathcal{T}[\mathcal{L}]$, $\mathcal{S}$, and $\mathcal{C}[\mathcal{L}]$, the input of $\mathcal{F}$ is

$$\left((\mathcal{S} \oplus \mathcal{T}[\mathcal{L}]) \parallel \mathcal{C}[\mathcal{L}] \parallel 0^* \parallel \mathcal{L}_T \parallel \beta + 5 \right).$$

We note here that here we do not use the byte-aligned assumption in our analysis. The analysis can be restricted to a byte-aligned one by adding a restriction on the families to have some of the values to be divisible by 8. Here the values that are related to the remainder of any value divided by 256 must be divisible by 8. So for example, in the case when $[\mathcal{L}_K] = 2$, if $\mathcal{L}_K = (\mathbf{a} \parallel \mathbf{b})$, then we do not need $\mathbf{a}$ to be divisible by 8. We just need $\mathbf{b}$ to be divisible by 8. This will not change the existence of any of the families. However, it will certainly requires a bigger parameter value. For example, for the case when $[\mathcal{L}_{K_1}] = [\mathcal{L}_{V_1}] = [\mathcal{L}_{K_2}] = 2$ and $[\mathcal{L}_{V_2}] = 1$, if we want all the values to be byte aligned, the smallest parameters we can use is when $\mathcal{L}_{K_1} = \mathcal{L}_{K_2} = 256$, $\mathcal{L}_{V_2} = 8$, and $\mathcal{L}_{V_1} = 2048$. This leads to the input size for the hash function to be at least $2048 + 256 + 5 \times 8 = 2394$ bits. Here we set $\mathcal{V}_2 = 128$ and $\mathcal{V}_1 = \mathcal{V}_2 \parallel 0^{2040}$ and the other settings to be the same as the previous example.

We move on to the tag generation when $\beta + 5$ changes from 1-byte value to 2-byte value. Note that the only possible source of this 1-byte value is from $\mathcal{L}_T$. So, in the second instantiation where $\beta + 5$ is changed to a 2-byte value, the tag length will be different from the initial one. In fact, the first tag length needs to be a 2-bytes value, say $a \parallel b$ and the second tag length needs to be a while the difference between the two β s needs to be $256 \times b$.

5.3.2 Modification of α

This section discusses a special case of the analysis in which the user has at least two instantiations of COFFE where they have different message block sizes but the same (or related) key. Since we are considering changing α , the one we really need to take care of is just the generation of $\mathcal{C}[0]$. This is because for any other place where α affects the system, it is generated by the previous chain in which we can truncate

easily.

Recall that $\mathcal{C}[0]$ is the first $\frac{\alpha}{4}$ post decimal values of π interpreted as hexadecimal values. Suppose that we want the difference of the two block sizes, α_1 and α_2 , to be k with α_2 being the larger value. Since we are assuming the ideality of $\mathcal{F}$, we want the input of $\mathcal{F}$ in this point for both instantiation to coincide. So in other words, if the initial vector of the first instantiation is the α_1 bit $\mathcal{C}_1$ and the second one to be the α_2 bit $\mathcal{C}_2$, the additional k bits of α_2 should be absorbed by the next substring of the input, which is the zero padding. Hence, the last k bits of $\mathcal{C}_2$ should all be zeros. In other words, the value of α_1 so that it can coincide with the positions in the post decimal values of π to have a consecutive 0^k bits. So for example, if we want $\alpha_2 = \alpha_1 + 8$, and α_1 and α_2 to be a multiple of 8, then we will need to wait until the 306-th decimal place to get the 8 bits of consecutive zeros. In this case, $\alpha_1 = 1224$ and $\alpha_2 = 1232$. The requirement that α_1 and α_2 are divisible by 8 comes if we are assuming that the design is byte-aligned. Note that different α s can be found along the places where we can find k consecutive zeros in the binary representation of π .

5.4 Distinguishing Attack

In this section, to form a distinguishing attack, we use the session key collision discussed in the previous section and the appendix. Assuming that we have the same secret key, different nonce, and different number of byte of domain value but the same session key, as before, we assume that now the session key for each instantiation is the same, each uses the proper number of byte of domain value.

We set the parameters $\alpha, \beta_1, \beta_2, \mathcal{L}_{T_1}$ and $\mathcal{L}_{T_2}$ as follows:

1. $\beta_1 + 5 < 256 < \beta_2 + 5 \leq \alpha + 5 \leq \delta + 5$
2. $\beta_2 - \beta_1 = 256\rho$ for some positive integer ρ
3. $\mathcal{L}_{T_1} < 256 \leq \mathcal{L}_{T_2}$ where $\mathcal{L}_{T_2} = 256\mathcal{L}_{T_1} + \rho$.

Set the first plaintext to be a two-blocks message, $\mathcal{M}_1 = (\mathcal{MB}_1 \parallel \mathcal{MB}_2)$ such that $\mathcal{MB}_1$ has α bits and $\mathcal{M}_2$ has β_1 bits with tag length set to be $\mathcal{L}_{T_2}$. Assume the ciphertext is $\mathcal{C}_1 = (\mathcal{CB}_1 \parallel \mathcal{CB}_2)$ with $\mathcal{T}_1$ as the tag.

The second message block, is then chosen to be $\mathcal{M}_2 = (\mathcal{MB}_1 \parallel \mathcal{MB}'_2)$ such that $\mathcal{MB}'_2$ has β_2 bits. Here we will use the tag length to be $\mathcal{L}_{T_1}$. We also assume the ciphertext is $\mathcal{C}_2 = (\mathcal{CB}'_1 \parallel \mathcal{CB}_2)$ with $\mathcal{T}_2$ as the tag.

As discussed above, since the first block of both message are the same, $\mathcal{MB}_1$, we should have $\mathcal{CB}_1 = \mathcal{CB}'_1$ and $\mathcal{T}[1]$ and $\mathcal{T}[2]$ should also be the same. Now remember that $\mathcal{MB}_2 \oplus \mathcal{CB}_2$ and $\mathcal{MB}'_2 \oplus \mathcal{CB}_2$ tells us the last β_1 and β_2 bits of $\mathcal{T}[2]$ respectively. So if $\mathcal{C}_1$ and $\mathcal{C}_2$ are both from COFFE instantiation, we must have $\mathcal{CB}_1 = \mathcal{CB}'_1$ and the last β_1 bits of $\mathcal{CB}_2$ and the last β_1 bits of $\mathcal{CB}'_2$ should agree. So we will guess that it is a COFFE instantiation instead of a random function if these requirements are met. Note that this can happen if it is a random function with probability $2^{-(\alpha+\beta_1)}$.

Recall that a distinguishing attack works as follows. An oracle randomly chooses whether it uses a random function or a COFFE instantiation with the given parameter. Then as an attacker, we can request for encryption for some plaintext. Then an adversary tries to decide whether the oracle uses a random function or a COFFE instantiation. The distinguishing attack described above has error probability 0 if we conclude that the oracle uses a random function. However, if we guess that the oracle uses a COFFE instantiation, there is a probability of $2^{-(\alpha+\beta_1)}$ of the function is actually a random function instead of COFFE. Note that since $\alpha \geq 256$ in our attack, the failure probability is at most 2^{-256} which is very small. Therefore, with 2 enquiries to the oracle with 4 message blocks, COFFE with ideal underlying hash function in nonce-respecting scenario can be distinguished with probability close to 1. So in these instantiations of COFFE, the security claim given in Lemma 5.2.1 is not satisfied.

5.5 Ciphertext Forgery Attack

In this section, we will propose two different ciphertext forgery attacks. The first attack is based on the observation on Subsection 5.3.1. It exploits the possibility of having an identical intermediate state value for two different instantiations when we fix α while using different values of β . The detail of the attack can be found in Section 5.5.1. Similarly, Section 5.5.2. discusses the forgery attack based on the discussion on Subsection 5.3.2. Here we try to forge a valid ciphertext in the case when there exists two different COFFE instantiations with same key for different message block size. Here both attacks require 2 enquiries and can forge a valid ciphertext with probability one. The success probability 1 is applicable whenever we assume for both instantiations, the secret key used is the same instead of one key being an extension of the other. Lastly, we will also discuss the possibility of combining the two forgery attacks. This can be found in Subsection 5.5.3.

5.5.1 Forgery Attack with Constant Message Block Size

Take any $(\mathcal{K}_1, \mathcal{V}_1), (\mathcal{K}_2, \mathcal{V}_2)$ (key, nonce) pairs we obtained in Section 5.3 such that they generate the same session key, one with 1-byte domain value, the other with two. Let $\mathcal{S}_1$ be the input for session key generation with 1-byte domain value and $\mathcal{S}_2$ be the input for the session key generation with 2-bytes domain value. Here we assume that the input for session key generation is chosen accordingly based on the number of bytes of domain value. Hence, after this point, we can ignore the secret key and nonce and we can just assume that for each instantiation, we are using the same session key and associated data.

Note that any full block plaintext-ciphertext pair leaks α least significant bits of the output of the hash function for a fixed input, while any β -bit block plaintext-ciphertext pair leaks only β least significant bits of it. So since $\beta \leq \alpha$, it is always more desirable to get a full-block plaintext-ciphertext pairs since they leak more bits

of the message digest.

Here we set the parameters $\alpha, \beta_1, \beta_2, \mathcal{L}_{T_1}$ and $\mathcal{L}_{T_2}$ as described before in Section 5.4.

Next we define the first message $\mathcal{M}_1$, a 3-block message $(\mathcal{MB}_1 \parallel \mathcal{MB}_2 \parallel \mathcal{MB}_3)$ such that $|\mathcal{MB}_1| = |\mathcal{MB}_2| = \alpha, |\mathcal{MB}_3| = \beta_2$. Let the ciphertext of this message be $\mathcal{C}_1 = (\mathcal{CB}_1 \parallel \mathcal{CB}_2 \parallel \mathcal{CB}_3)$ with tag $\mathcal{T}_1$ with $\mathcal{L}_T$ set to any value. Since $\mathcal{CB}_1$ and $\mathcal{CB}_2$ are known, $\mathcal{MB}_1 \oplus \mathcal{CB}_1$ gives us the last α bits of $\mathcal{T}[1]$ and $\mathcal{M}_2 \oplus \mathcal{C}_2$ gives us the last α bits of $\mathcal{T}[2]$ which are essential in the attack.

We define our second message $\mathcal{M}_2$, a 2-block message $(\mathcal{MB}_1 \parallel \mathcal{MB}'_2)$ with the length of $\mathcal{MB}'_2$ to be β_1 bits and tag length to be $\mathcal{L}_{T_2}$. The first block is chosen to be exactly the same as before to ensure the value of $\mathcal{T}[1]$ and $\mathcal{T}[2]$ can be kept constant. Based on the previous message, the least α bits of both values are known. Suppose that the ciphertext of this plaintext is $\mathcal{C}_2 = (\mathcal{CB}_1 \parallel \mathcal{CB}'_2)$ with tag $\mathcal{T}_2$.

Using the information we obtain, we generate the following valid ciphertext. Define another 2-block message $\mathcal{M}_3 = (\mathcal{MB}_1 \parallel \mathcal{MB}''_2)$. Here we set $\mathcal{MB}_1$ to be the same as the first block from the previous message blocks. This is again to ensure the value of $\mathcal{T}[1]$ and $\mathcal{T}[2]$ can be kept constant. We let the length of $\mathcal{MB}''_2$ to be β_2 and choose $\mathcal{MB}''_2$ such that $\mathcal{MB}''_2 \oplus_{\beta_2} \mathcal{T}[2] = \mathcal{C}_2 \parallel 0^{\beta_2 - \beta_1}$. Here, $\mathcal{MB}''_2$ can be calculated since we know the last α bits of $\mathcal{T}[2]$ and $\alpha > \beta_2$. Using this message, it is easy to see that the tag generation will have the same input as before, although $\mathcal{L}_T$ is now $\mathcal{L}_{T_1}$. So the tag for this ciphertext will be the last β_1 bits of $\mathcal{T}_2$.

This attack has success probability equal to the probability of the two strings used as the input session key generation to be the same. As we have discussed before, for some parameters such as the ones in case I.3 and case II.3, this can even be 1. In other words, in the case when the success probability is one, the attack above proves that the ciphertext integrity of this cipher does not satisfy the bound given in lemma 5.2.2 even in an ideal hash function situation.

Note that here we use three COFFE instantiations for each attack (2 for enquiry and 1 for the guess), while in our discussion on session key generation collision, we only

consider the collision for two (key, nonce) pairs. So the same attack cannot directly work for nonce-respecting scenario unless we can find three (key, nonce) pairs that collide to the same session key.

5.5.2 Forgery Attack with Dynamic Message Block Size

In this section, we are assuming the existence of two different instantiations of COFFE with different message block size but the same secret key and constant last message block size β . Now we pick $\alpha_1 < \alpha_2$ such that $\alpha_2 - \alpha_1 = k$. Next we find the valid size of α_1 and α_2 based on our discussion in the discussion section. Here since we assume constant last message block size, β , we assume $\beta \leq \alpha_1$. Since we are using constant last message block size, to get the same session key, we can consider the nonce-misuse scenario where we use the same key and nonce for both instantiations. Note that this means the tag length should still be kept the same.

First, we generate message, $\mathcal{M}_1 = (\mathcal{MB}_1 \parallel \mathcal{MB}_2)$ with $|\mathcal{MB}_1| = \alpha_2$ and $|\mathcal{MB}_2| = \beta$. Now assume that we get the ciphertext $\mathcal{C}_1 = (\mathcal{CB}_1, \mathcal{CB}_2)$ with tag $\mathcal{T}_1$. In this pair, our objective is to find the last α_2 bits of $\mathcal{T}[1]$ which can be obtained by calculating $\mathcal{MB}_1 \oplus \mathcal{CB}_1$.

We then consider the following message: $\mathcal{M}_2 = (\mathcal{MB}'_1 \parallel \mathcal{MB}'_2)$ with $|\mathcal{MB}'_1| = \alpha_2$ and $|\mathcal{MB}'_2| = \beta$. We further require the last k bits of $\mathcal{MB}'_2 \oplus_{\alpha_1} \mathcal{T}[1]$ are all zeros. Note that $\mathcal{MB}'_2$ can be generated easily with the knowledge of the last α_1 bits of $\mathcal{T}[1]$. Assume that the ciphertext is $\mathcal{C}_2 = (\mathcal{CB}'_1 \parallel \mathcal{CB}'_2)$ with tag $\mathcal{T}_2$. Here the last k bits of $\mathcal{CB}'_1$ are all zero and $\mathcal{CB}'_2 \oplus \mathcal{MB}'_2$ tells us the last β bits of $\mathcal{T}[2]$ when the first block of the message is $\mathcal{MB}'_1$.

Now having this information, we will proceed to our forgery attack. The message block that we will use is $\mathcal{M}_3 = (\mathcal{MB}''_1 \parallel \mathcal{MB}'_2)$. Here we have $|\mathcal{MB}''_1| = \alpha_1$ and $\mathcal{MB}''_1$ is chosen such that

$$(\mathcal{MB}''_1 \oplus_{\alpha_1} \mathcal{T}[1]) \parallel 0^k = \mathcal{CB}'_1.$$

We also note that the second block is exactly the same used in $\mathcal{M}_2$. Then it is

easy to see that the ciphertext is $\mathcal{C}_3 = (\mathcal{CB}_1'' \parallel \mathcal{CB}_2')$ where $\mathcal{CB}_1'' = \mathcal{MB}_1'' \oplus_{\alpha_1} \mathcal{T}[1]$. Furthermore, the tag is exactly $\mathcal{T}_2$.

This forgery attack requires 2 enquiries to the oracle with 4 message blocks. So this attack provides a family of instances of COFFE that cannot provide ciphertext integrity as claimed in Lemma 5.2.2 under nonce-misuse scenario.

5.5.3 Combination of the Existing Attacks

In our previous two subsections, we change one of the parameters (α, β) while letting the other constant. This is done to simplify the analysis. However, it is possible for us to combine both attacks to generate new attack, that is, we change α and β in the same time. Notice that by combining the two attacks, the “nonce-misuse” requirement is not a must anymore. As discussed in the constant message block size subsection, as long as we can find a triple of (key, nonce) pairs that generate the same session key, we can launch the attack in the nonce-respecting scenario.

5.6 Related Key Recovery Attack

Note that, in most of the attacks we have mentioned, we are assuming same secret key. In this section, we will discuss the case with two different instances of COFFE with different key length. As discussed in our observations, in this case we assume that the longer key is obtained by extending the shorter key with secret string. As we have discussed in the appendix, there are $(\mathcal{K}_1, \mathcal{V}_1), (\mathcal{K}_2, \mathcal{V}_2)$ pairs that leads to the same session key (with one of them using one-byte domain value while the other using two-byte value) with different key length. Here we will use the pairs with k -bits key length difference and all of the difference are all in the nonce of the corresponding shorter length key. Now assume that $|\mathcal{L}_{K_1}| > |\mathcal{L}_{K_2}|$.

We again choose the parameters $\alpha, \beta_1, \beta_2, \mathcal{L}_{T_1}$, and $\mathcal{L}_{T_2}$ as in Section 5.4. The attack here is an adaptation of the distinguishing attack we proposed earlier. We

use the two messages $\mathcal{M}_1$ and $\mathcal{M}_2$ as described in section 4. The difference here is that for $\mathcal{M}_2$ with two bytes domain value and shorter key length, we will enquire 2^k different blocks of it with different k most significant bits of $\mathcal{V}_2$. Note that if the k most significant bits of $\mathcal{V}_2$ coincide with the k -bit extension of the secret key, then $\mathcal{CB}_1 = \mathcal{CB}'_1$ and the last β_1 bits of $\mathcal{CB}_2$ and the last β_1 bits of $\mathcal{CB}'_2$ should agree. So by using this approach, we can guess the k -bits extension of the secret key with the same complexity as exhaustive search for a k -bits secret key.

As discussed in the distinguishing attack section, when we decide that the guessed k -bits is wrong, the probability that the k -bits is actually the correct extension key is 0. So there will not be a false negative. However, when the k -bits we guess is wrong, the probability of false positive is, as discussed in the distinguishing attack, $2^{\alpha+\beta_1}$ which is at least 2^{-255} which is negligible.

So for the related key recovery attack, to recover the k -bit extension of the secret key in nonce-respecting scenario, we will need $2^k + 1$ plaintext-ciphertext pairs with success probability approximately 1. Note that the exact same attack can be adapted to the case when $|\mathcal{K}_1| < |\mathcal{K}_2|$.

5.7 Conclusion

5.7.1 Attack Summary

From the discussion above, we see that the security claim for the nonce-misusing scenario is not met for many different parameters. The same attack can be adapted to give a distinguishing attack for the nonce-respecting scenario for some subfamilies of the parameters mentioned above.

Lastly, having two different instances of COFFE with different key length with the longer key being the extension of the shorter key may not be a good idea. This is because if the parameter used belongs to the family we have found earlier, the extension of the key can be recovered with exhaustive search in the same way as if

the secret key is just k bits where k is the length of the extension of the secret key.

In conclusion, COFFE does not satisfy any of the two security claims for some of the parameters that we have discussed before. The problem arises from the fact that concatenation of strings cannot be inverted uniquely and hence giving the opportunity of having two different set of strings concatenated to the same resulting strings.

5.7.2 Lesson Learned

Here we see that the forgery and distinguishing attacks are feasible due to the possibility to have different (key,nonce) pairs to generate the same session key. This can be fixed by fixing the space for every given parameters. If some parameters are variables (such as the message blk size in COFFE), we should ensure that the values of the variables get authenticated so as to prevent the forgery attack.

Chapter 6

Conclusion

In this thesis, we have considered the security of some symmetric key cryptosystems. We briefly review some basic concept on symmetric key cryptosystems and several basic cryptanalysis methods which focus mainly on differential cryptanalysis and its variants. We then apply the attacks to Generalized Feistel scheme and two authenticated encryption schemes.

We applied truncated differential analysis, impossible differential cryptanalysis and Boomerang attack to analyse a specific block cipher design, Simpira v2 with 3 branches. In this, we exploited the asymmetry of its encryption and decryption in which the diffusion of the decryption is relatively slower. Because of this, a differential path (more specifically, a truncated differential path) can pass through more rounds with probability still high enough for the attack to be launched. This shows us that when designing a Feistel-based or Generalized Feistel-based cryptosystem, we need to make sure that both encryption and decryption are secure against attacks. To the best of our knowledge, this is the first analysis on this specific variant of Simpira v2. This study shows that the security margin provided by the designer of Simpira v2 is good. However, it also shows that aside from analyzing the forward direction of the cipher, we also need to consider the backward direction since in some designs, the different directions may provide different diffusion capability.

We then studied five types of Generalized Feistel Schemes. In this analysis, we assume that the round functions are ideal random functions. We found that in GFS type 1 with k branches, with the same complexity, we can reach $k - 1$ more rounds than the analysis on encryption of the scheme due to the asymmetry of the design. This is not true in GFS type 2 and 3 where the encryption and decryption are more symmetric. This can be seen that in these two types, both analysis on encryption and decryption reach approximately the same number of rounds. In the Alternating Feistel Scheme, a better lower bound can be found in almost all the cases, most evident in the case when the round number is odd. Lastly, we also consider two cases of $UFN(k', k)$ where either k' or $k - k'$ is a factor of k . To the best of our knowledge, this is the first general analysis on these cases where k' and k can have any values as long as either k' or $k - k'$ is a factor of k . Since this attack applies for any round function, the attacks discussed in this chapter are applicable to any design using the Generalized Feistel Network described above. So this study is helpful in the security analysis of cryptosystem designs that use Generalized Feistel Network as their structures.

In the last half of the thesis, we analysed two authenticated ciphers and their security. Firstly, we analysed the security of ICEPOLE. In our analysis, we considered the assumption that both public nonce and secret message number can be repeated. Under this condition, we found that the state value can be recovered by using the differential-linear analysis. In the case when secret message number is used, although we cannot recover the secret key, we can invert the state value we found until the state where only the secret message number has been processed. Using this information, we can validly encrypt any plaintext that we want and we will get a valid tag. The lesson that we have learned from this is that in the authenticated encryption design, when we want the design to be secure against the nonce-reuse assumption, we need to be careful in analysing the security of the round function. Furthermore, it will be better if the analysis is done to the permutation although some of the state cannot

be controlled in the practical attack. This may provide a better security guarantee and it will be more secure.

Lastly, we analysed the security of a hash-based authenticated encryption scheme COFFE. Instead of finding an input difference that can go to an output difference with big enough probability, we consider if it is possible for two distinct inputs to lead to zero difference in the state before encryption. In some cases, this can be achieved even when the underlying hash function is assumed to be an ideal hash function. So we learn that when we allow an authenticated encryption scheme design to have different parameter values, we need to make sure that the parameter values along with their length in their binary representation is also authenticated. Otherwise, it is possible to exploit different parameter values to either distinguish the encryption from a random permutation or to even recover some of the secret key.

List of Publications

1. Tao Huang, Ivan Tjuawinata, and Hongjun Wu. Differential-Linear Cryptanalysis of ICEPOLE, *Fast Software Encryption: 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, pages 243 - 263. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.
2. Ivan Tjuawinata, Tao Huang, and Hongjun Wu. Cryptanalysis of the Authenticated Encryption Algorithm COFFE, *Selected Areas in Cryptography – SAC 2015: 22nd International Conference, Sackville, NB, Canada, August 12–14, 2015, Revised Selected Papers* pages 510 - 526. Springer International Publishing, Cham, 2016.
3. Ivan Tjuawinata, Tao Huang, and Hongjun Wu. Cryptanalysis of Simpira v2, *Information Security and Privacy: 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3–5, 2017, Proceedings, Part I* pages 384 - 401. Springer International Publishing, Cham, 2017.

Accepted Papers

1. Ivan Tjuawinata, Tao Huang, and Hongjun Wu. Improved Differential Cryptanalysis on Generalized Feistel Schemes. (Accepted in INDOCRYPT 2017)

I played an active role in finding the results and writing the following papers:

- Cryptanalysis of the Authenticated Encryption Algorithm COFFE.
- Cryptanalysis of Simpira v2.
- Improved Differential Cryptanalysis on Generalized Feistel Schemes.

Differential-Linear Cryptanalysis of ICEPOLE was mainly done by Tao Huang.

Bibliography

- [1] 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, 3G Security, Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 2: KASUMI Specification, V.3.1.1. 2001.
- [2] "The 128-bit Blockcipher CLEFIA Algorithm Specification,". CLEFIA Standardization in ISO/IEC 29192-2 (Online), 2007.
- [3] Secure Hash Standard (SHS). 2008.
- [4] ISO/IEC 19772:2009. Information technology - Security techniques - Authenticated Encryption. 2009.
- [5] SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions . 2014.
- [6] A Mathematical Theory of Cryptography. September 1, 1945.
- [7] 3GPP. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3, Document 1, 128-EEA3 and 128-EIA3 specification. The 3rd Generation Partnership Project (3GPP), 2010.
- [8] D. Eastlake 3rd and P. Jones. US Secure Hash Algorithm 1 (SHA1). September 2001.
- [9] Ross Anderson, Eli Biham, and Lars Knudsen. Serpent: A Proposal for the Advanced Encryption Standard. NIST AES Proposal, 1998.

- [10] Kazumaro Aoki, Tetsuya Ichikawa, Masayuki Kanda, Mitsuru Matsui, Shiho Moriai, Junko Nakajima, and Toshio Tokita. Camellia: a 128-bit block cipher suitable for multiple platforms - design and analysis. In *D.R. Stinson and S. Tavares (Eds.) SAC 2000*, volume 2012 of *Lecture Notes in Computer Science*, pages 39–56. Springer Heidelberg, 2001.
- [11] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C-W Phan. SHA-3 Proposal BLAKE. Submission to NIST (2010).
- [12] Jean-Philippe Aumasson, Philipp Jovanovic, and Samuel Neves. NORX v3.0. Submission to CAESAR. Available from: <http://competitions.cr.yp.to/caesar-submissions.html>, 2016.
- [13] William Curt Barker. *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher (Revised January 2012)*. US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 2012.
- [14] Guy Barwell. Forgery on Stateless CMCC with a Single Query. 2014.
- [15] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. pages 1–15. Springer-Verlag, 1996.
- [16] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology – ASIACRYPT 2000*, pages 531 – 545, 2000.
- [17] Côme Berbain, Olivier Billet, Anne Canteaut, Nicolas Courtois, Henri Gilbert, Louis Goubin, Aline Gouget, Louis Granboulan, Cédric Lauradoux, Marine Minier Marine, et al. Sosemanuk, A Fast Software-Oriented Stream Cipher. In *New Stream Cipher Designs*, pages 98–118. Springer, 2008.

- [18] Daniel J. Bernstein. Failures of Secret-Key Cryptography. Invited talk in FSE 2013, available from <http://cr.yp.to/talks/2013.03.12/slides.pdf>.
- [19] Daniel J. Bernstein. Chacha, a variant of salsa 20. In Jean Jacques Quisquater and Bruce Schneier, editors, *SASC 2008 – The State of the Art of Stream Ciphers*. *ECRYPT*, 2008.
- [20] Daniel J. Bernstein. The Salsa20 Family of Stream Ciphers. In *New Stream Cipher Designs*, pages 84–97. Springer, 2008.
- [21] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak Sponge Function Family Main Document. Submission to NIST (Round 2), 2009.
- [22] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the sponge: Single-pass authenticated encryption and other applications. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography*, volume 7118 of *Lecture Notes in Computer Science*, pages 320–337. Springer Berlin Heidelberg, 2012.
- [23] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. CAESAR submission: Ketje v2. Submission to CAESAR. Available from: <http://competitions.cr.yp.to/caesar-submissions.html>, 2016.
- [24] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. CAESAR submission: Keyak v2. Submission to CAESAR. Available from: <http://competitions.cr.yp.to/caesar-submissions.html>, 2016.
- [25] Eli Biham, Alex Biryukov, and Adi Shamir. *Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials*, pages 12–23. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

- [26] Eli Biham, Orr Dunkelman, and Nathan Keller. *Enhancing Differential-Linear Cryptanalysis*, pages 254–266. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.
- [27] Eli Biham, Orr Dunkelman, and Nathan Keller. Differential-Linear Cryptanalysis of Serpent. In Thomas Johansson, editor, *Fast Software Encryption*, volume 2887 of *Lecture Notes in Computer Science*, pages 9–21. Springer Berlin Heidelberg, 2003.
- [28] Eli Biham, Orr Dunkelman, and Nathan Keller. *A Related-Key Rectangle Attack on the Full KASUMI*, pages 443–461. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.
- [29] Eli Biham and Adi Shamir. *Differential Cryptanalysis of DES-like Cryptosystems*, volume 4, pages 3–72. Springer-Verlag, 1991.
- [30] Eli Biham and Adi Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer-Verlag, London, UK, 1993.
- [31] Begül Bilgin, Andrey Bogdanov, Miroslav Knežević, Florian Mendel, and Qingju Wang. Fides: Lightweight Authenticated Cipher with Side-Channel Resistance for Constrained Hardware. In *Cryptographic Hardware and Embedded Systems-CHES 2013*, pages 142–158. Springer, 2013.
- [32] Alex Biryukov. A new 128-bit key stream cipher LEX. *eSTREAM, ECRYPT Stream Cipher Project, Report*, 13:2005, 2005.
- [33] Alex Biryukov. *The Boomerang Attack on 5 and 6-Round Reduced AES*, pages 11–15. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.
- [34] Alex Biryukov. *Feistel Cipher*, pages 455–455. Springer US, Boston, MA, 2011.

- [35] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolić. *Distinguisher and Related-Key Attack on the Full AES-256*, pages 231–249. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [36] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway. *UMAC: Fast and Secure Message Authentication*, pages 216–233. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.
- [37] John Black and Phillip Rogaway. *CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions*, pages 197–215. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.
- [38] John Black and Phillip Rogaway. A Suggestion for Handling Arbitrary-Length Messages with the CBC MAC. Natl. Inst. Stand. Technol., <http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/workshop1/papers/black-rogoway-xcbc.pdf>, 2001.
- [39] John Black and Phillip Rogaway. *A Block-Cipher Mode of Operation for Parallelizable Message Authentication*, pages 384–397. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.
- [40] Céline Blondeau, Gregor Leander, and Kaisa Nyberg. Differential-Linear Cryptanalysis Revisited. In *Fast Software Encryption–FSE 2014*. Springer, 2014.
- [41] Martin Boesgaard, Mette Vesterager, Thomas Pedersen, Jesper Christiansen, and Ove Scavenius. Rabbit: A New High-Performance Stream Cipher. In *Fast Software Encryption*, pages 307–329. Springer, 2003.
- [42] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsøe. *PRESENT: An Ultra-Lightweight Block Cipher*, pages 450–466. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

- [43] Andrey Bogdanov, Florian Mendel, Francesco Regazzoni, Vincent Rijmen, and Elmar Tischhauser. ALE: AES-Based Lightweight Authenticated Encryption. In *Fast Software Encryption*, 2013.
- [44] CAESAR. Competition for Authenticated Encryption: Security, Applicability, and Robustness. <http://competitions.cr.yp.to/caesar.html>.
- [45] Simon Cogliani, Diana Ștefania Maimuț, David Naccache, Rodrigo Portella do Canto, Reza Reyhanitabar, Serge Vaudenay, and Damian Vizár. Offset Merkle-Damgård (OMD) version 1.0: A CAESAR Proposal. 2014.
- [46] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. *Merkle-Damgård Revisited: How to Construct a Hash Function*, pages 430–448. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.
- [47] Paul Crowley. Truncated differential cryptanalysis of five rounds of salsa20. 2006.
- [48] Joan Daemen and Vincent Rijmen. The block cipher rijndael. In Jean Jacques Quisquater and Bruce Schneier, editors, *Smart Card Research and Applications, LNCS*, volume 1820, pages 277–284. Springer Berlin Heidelberg, 2000.
- [49] Joan Daemen and Vincent Rijmen. The Wide Trail Design Strategy. In Bahram Honary, editor, *Cryptography and Coding*, volume 2260 of *LNCS*, pages 222–238. Springer Berlin Heidelberg, 2001.
- [50] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES—the Advanced Encryption Standard*. Springer, 2002.
- [51] Joan Daemen and Vincent Rijmen. Understanding Two-Round Differentials in AES. In *International Conference on Security and Cryptography for Networks*, pages 78–94. Springer, 2006.
- [52] N. Datta and M. Nandi. ELmD v1.0. Submission to CAESAR. 2014.

- [53] Itai Dinur and Jeremy Jean. Cryptanalysis of FIDES. In *Fast Software Encryption–FSE 2014*. Springer, 2014.
- [54] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Cryptanalysis of Simpira v1. Cryptology ePrint Archive, Report 2016/244, 2016. <http://eprint.iacr.org/2016/244>.
- [55] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schlffer. Ascon v1.2. Submission to CAESAR. Available from: <http://competitions.cr.yp.to/caesar-submissions.html>, 2016.
- [56] Orr Dunkelman, Sebastiaan Indesteege, and Nathan Keller. A Differential-Linear Attack on 12-Round Serpent. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *Progress in Cryptology - INDOCRYPT 2008*, volume 5365 of *Lecture Notes in Computer Science*, pages 308–321. Springer Berlin Heidelberg, 2008.
- [57] Orr Dunkelman and Nathan Keller. A New Attack on the LEX Stream Cipher. In *ASIACRYPT 2008*, pages 539–556. Springer, 2008.
- [58] Orr Dunkelman and Nathan Keller. Cryptanalysis of etc2. In Marc Fischlin, editor, *Topics in Cryptology CT–RSA 2009*, volume 5473 of *Lecture Notes in Computer Science*, pages 226–239. Springer Berlin Heidelberg, 2009.
- [59] Orr Dunkelman, Nathan Keller, and Adi Shamir. Minimalism in Cryptography: The Even-Mansour Scheme Revisited. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 336–354, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- [60] T. Duong and J. Rizzo. Here Come the $\oplus$ Ninjas. 2011.

- [61] M. Dworkin. Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. May 2004.
- [62] M. Dworkin. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. November 2007.
- [63] Daniel Engels, Markku-Juhani O. Saarinen, Peter Schweitzer, and Eric M. Smith. The hummingbird-2 lightweight authenticated encryption algorithm. In *RFIDSec 2011*, pages 19 – 31. Springer, 2011.
- [64] Shimon Even and Yishay Mansour. A Construction of a Cipher from a Single Pseudorandom Permutation. *Journal of Cryptology*, 10(3):151–161, 1997.
- [65] Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, and Jesse Walker. The Skein Hash Function Family. Submission to NIST.
- [66] Niels Ferguson, Doug Whiting, Bruce Schneier, John Kelsey, Stefan Lucks, and Tadayoshi Kohno. Helix, fast encryption and authentication in a single cryptographic primitive. In *Fast Software Encryption - FSE 2003*, pages 330 – 346. Springer, 2003.
- [67] Christian Forler, David McGrew, Stefan Lucks, and Jakob Wenzel. Coffe: Ciphertext output feedback faithful encryption. In *Early Symmetric Crypto ESC 2013*, page 6, 2013.
- [68] Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schlffer, and Sren S. Thomsen. Grstl–a SHA-3 candidate. Submission to NIST, 2008.
- [69] Marc Girault, Robert Cohen, and Mireille Campana. *A Generalized Birthday Attack*, pages 129–156. Springer Berlin Heidelberg, Berlin, Heidelberg, 1988.

- [70] Shay Gueron and Nicky Mouha. Simpira: A Family of Efficient Permutations Using the AES Round Function. Cryptology ePrint Archive, Report 2016/122, 2016. <http://eprint.iacr.org/2016/122>.
- [71] Shay Gueron and Nicky Mouha. Simpira v2: A Family of Efficient Permutations Using the AES Round Function. In *Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I 22*, pages 95–125. Springer, 2016.
- [72] Jian Guo, Thomas Peyrin, Axel Poschmann, and Matt Robshaw. The LED Block Cipher. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems–CHES 2011*, volume 6917 of *LNCS*, pages 326–341. Springer Berlin Heidelberg, 2011.
- [73] B. Preneel H. Wu. AEGIS: A Fast Authenticated Encryption Algorithm (v1.1). 2014.
- [74] Viet Tung Hoang and Phillip Rogaway. On generalized feistel networks. In *Advances in Cryptology – CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010*. Springer Berlin Heidelberg, 2010.
- [75] Viet Tung Hoang and Phillip Rogaway. *On Generalized Feistel Networks*, pages 613–630. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
- [76] Seokhie Hong, Sangjin Lee, Jongin Lim, Jaechul Sung, Donghyeon Cheon, and Inho Cho. Provable Security against Differential and Linear Cryptanalysis for the SPN Structure. In Gerhard Goos, Juris Hartmanis, Jan Leeuwen, and Bruce Schneier, editors, *Fast Software Encryption*, volume 1978 of *LNCS*, pages 273–283. Springer Berlin Heidelberg, 2001.

- [77] ISO. ISO/IEC 9797-1:1999 information technology – security techniques – message authentication codes (MACs) – part 1: Mechanisms using a block cipher. Technical report, International Organization for Standardization, 1999.
- [78] ISO/IEC 19772:2009. *Information technology – Security techniques – Authenticated encryption*. ISO, Geneva, Switzerland, 2009.
- [79] T. Iwata and K. Kurosawa. OMAC: One-key CBC MAC. In *Fast Software Encryption*, volume 2887 of *LNCS*, pages 129–153. Springer, 2003.
- [80] Goce Jakimoski and Samant Khajuria. Asc-1 : An authenticated encryption stream cipher. In *Selected Areas in Cryptography - SAC 2011*, pages 356 – 372. Springer, 2011.
- [81] Charanjit S. Jutla. Generalized birthday attacks on unbalanced feistel networks. In *in proceedings of Crypto’98, LNCS*, pages 186–199. Springer-Verlag, 1998.
- [82] C.S. Jutla. Encryption modes with almost free message integrity. In *Advances in Cryptology – EUROCRYPT 2001*, pages 529 – 544. Springer, 2001.
- [83] Lars Knudsen. Deal - a 128-bit block cipher. In *NIST AES Proposal*, 1998.
- [84] Lars R. Knudsen. *Truncated and Higher Order Differentials*, pages 196–211. Springer Berlin Heidelberg, Berlin, Heidelberg, 1995.
- [85] Lars R. Knudsen, M. J. B. Robshaw, and David Wagner. *Truncated Differentials and Skipjack*, pages 165–180. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.
- [86] L.R. Knudsen and T.A. Berson. Truncated differentials of safer. In D. Gollman, editor, *Fast Software Encryption - Third International Workshop, LNCS 1039*, pages 15–25. Springer Verlag, 1996.

- [87] Hugo Krawczyk. *The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?)*, pages 310–331. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.
- [88] Ted Krovetz and Phillip Rogaway. The Software Performance of Authenticated-Encryption Modes. In *Fast Software Encryption – FSE 2011*, pages 306–327. Springer, 2011.
- [89] Susan K. Langford. *Differential-linear Cryptanalysis and Threshold Signatures*. PhD thesis, Stanford, CA, USA, 1995. UMI Order No. GAX95-35617.
- [90] Susan K. Langford and Martin E. Hellman. *Differential-Linear Cryptanalysis*, pages 17–25. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.
- [91] Seonhee Lee, Seokhie Hong, Sangjin Lee, Jongin Lim, and Seonhee Yoon. *Truncated Differential Cryptanalysis of Camellia*, pages 32–38. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.
- [92] Gaëtan Leurent and Thomas Fuhr. Observation on PiCipher. 2014.
- [93] Jiqiang Lu. A Methodology for Differential-Linear Cryptanalysis and Its Applications. 7549:69–89, 2012.
- [94] Jiqiang Lu, Orr Dunkelman, Nathan Keller, and Jongsung Kim. *New Impossible Differential Attacks on AES*, pages 279–293. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
- [95] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.*, 17(2):373–386, apr 1988.
- [96] Mitsuru Matsui. Linear Cryptanalysis Method for DES cipher. In *Advances in Cryptology–EUROCRYPT’93*, pages 386–397. Springer, 1994.

- [97] Mitsuru Matsui and Atsuhiro Yamagishi. A new method for known plaintext attack of feal cipher. In *Advances in Cryptology - EUROCRYPT'92*, pages 81–91. Springer, 1993.
- [98] David McGrew and John Viega. The Galois/Counter Mode of Operation (GCM). <http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-spec.pdf>.
- [99] C. Meyer and S. Matyas. Secure Program Load with Manipulation Detection Code. In *Proceedings of Securicom vol.88*, pages 111–130, 1988.
- [100] Bodo Möller, Thai Duong, and Krzysztof Kotowicz. This POODLE Bites: Exploiting the SSL 3.0 Fallback. September 2014.
- [101] Paweł Morawiecki. Nonce reuse in ICEPOLE. available from <http://competitions.cr.yp.to/round1/icepole-misuse.html>, Jul 2014.
- [102] Paweł Morawiecki, Kris Gaj, Ekawat Homsirikamol, Krystian Matusiewicz, Josef Pieprzyk, Marcin Rogawski, Marian Srebrny, and Marcin Wójcik. ICE-POLE v2. submission to CAESAR competition, available from <http://competitions.cr.yp.to/caesar-submissions.html>.
- [103] Paweł Morawiecki, Kris Gaj, Ekawat Homsirikamol, Krystian Matusiewicz, Josef Pieprzyk, Marcin Rogawski, Marian Srebrny, and Marcin Wójcik. ICE-POLE: High-Speed, Hardware-Oriented Authenticated Encryption. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems, CHES 2014*, volume 8731 of *Lecture Notes in Computer Science*, pages 392–413. Springer Berlin Heidelberg, 2014.
- [104] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In Chuan-Kun Wu, Moti Yung, and Dongdai Lin, editors, *Information Security and Cryptology*,

- volume 7537 of *Lecture Notes in Computer Science*, pages 57–76. Springer Berlin Heidelberg, 2012.
- [105] Valérie Nachev, Emanuel Volte, and Jacques Patarin. *Differential Attacks on Generalized Feistel Schemes*, pages 1–19. Springer International Publishing, Cham, 2013.
 - [106] Phuong Ha Nguyen, Matthew J. B. Robshaw, and Huaxiong Wang. *On Related-Key Attacks and KASUMI: The Case of A5/3*, pages 146–159. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
 - [107] I. Nikolić. Tiaoxin-346. 2014.
 - [108] NIST. Recommendation for Block Cipher Modes of Operation-Methods and Techniques. NIST special publication 800–38A, 2001 Edition.
 - [109] NIST. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. NIST special publication 800–38B, 2001 Edition.
 - [110] NIST. The Keyed-Hash Message Authentication Code (HMAC). FIPS PUB 198.
 - [111] Kaisa Nyberg. *Generalized Feistel Networks*, pages 91–104. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.
 - [112] National Institute of Standards and Technology. Data Encryption Standard (DES). 1999.
 - [113] Jacques Patarin. *Generic Attacks on Feistel Schemes*, pages 222–238. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.
 - [114] Jacques Patarin. Security of random feistel schemes with 5 or more rounds. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Pro-*

- ceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 106–122. Springer, 2004.
- [115] Jacques Patarin. Security of balanced and unbalanced feistel schemes with linear non equalities. *IACR Cryptology ePrint Archive*, 2010.
 - [116] Jacques Patarin, Valérie Nachev, and Côme Berbain. *Generic Attacks on Unbalanced Feistel Schemes with Contracting Functions*, pages 396–411. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.
 - [117] Jacques Patarin, Valérie Nachev, and Côme Berbain. *Generic Attacks on Unbalanced Feistel Schemes with Expanding Functions*, pages 325–341. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.
 - [118] Bart Preneel, René Govaerts, and Joos Vandewalle. *Hash functions based on block ciphers: a synthetic approach*, pages 368–378. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.
 - [119] Ronald L. Rivest. The MD5 Message-Digest Algorithm, Request for Comments (RFC 1320). 1992.
 - [120] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. Ocb: A block-cipher mode of operation for efficient authenticated encryption. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 196 – 205, 2001.
 - [121] Phillip Rogaway. Authenticated-Encryption with Associated-Data. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 98–107. ACM, 2002.
 - [122] Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes ocb and pmac. In *Advances in Cryptology – ASIACRYPT 2004*, pages 16 – 31, 2004.

- [123] Sondre Rønjom. Invariant Subspaces in Simpira. Cryptology ePrint Archive, Report 2016/248, 2016. <https://eprint.iacr.org/2016/248>.
- [124] Bruce Schneier. *Description of a new variable-length key, 64-bit block cipher (Blowfish)*, pages 191–204. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.
- [125] Bruce Schneier and John Kelsey. *Unbalanced Feistel networks and block cipher design*, pages 121–144. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.
- [126] Yongsup Shin, Jongsung Kim, Guil Kim, Seokhie Hong, and Sangjin Lee. Differential-Linear Type Attacks on Reduced Rounds of SHACAL-2. In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *Information Security and Privacy*, volume 3108 of *Lecture Notes in Computer Science*, pages 110–122. Springer Berlin Heidelberg, 2004.
- [127] Siang Meng Sim and Lei Wang. Practical Forgery Attacks on SCREAM and iSCREAM. 2014.
- [128] Douglas R. Stinson. *Cryptography Theory and Practice*. Chapman & Hall/CRC, 3 edition, 2006.
- [129] Kazuhiro Suzuki, Dongvu Tonien, Kaoru Kurosawa, and Koji Toyota. Birthday Paradox for Multi-collisions. In Minsurp Rhee and Byoungcheon Lee, editors, *Information Security and Cryptology ICISC 2006*, volume 4296 of *Lecture Notes in Computer Science*, pages 29–40. Springer Berlin Heidelberg, 2006.
- [130] Joana Treger and Jacques Patarin. *Generic Attacks on Feistel Networks with Internal Permutations*, pages 41–59. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [131] Emanuel Volte, Valérie Nachev, and Jacques Patarin. *Improved Generic Attacks on Unbalanced Feistel Schemes with Expanding Functions*, pages 94–111. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

- [132] David Wagner. *The Boomerang Attack*, pages 156–170. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.
- [133] Yuechuan Wei, Ping Li, Bing Sun, and Chao Li. *Impossible Differential Cryptanalysis on Feistel Ciphers with SP and SPS Round Functions*, pages 105–122. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
- [134] Doug Whiting, Russ Housley, and Niels Ferguson. Counter with CBC-MAC (CCM). Available from <http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmo-des/ccm/ccm.pdf>, 2003.
- [135] Doug Whiting, Bruce Schneier, Stefan Lucks, and Frédéric. Muller. Phelix : Fast Encryption and Authentication in a Single Cryptographic Primitive. 2005.
- [136] H. Wu and T. Huang. JAMBU Lightweight Authenticated Encryption Mode and AES-JAMBU. 2014.
- [137] H. Wu and T. Huang. JAMBU Lightweight Authenticated Encryption Mode and AES-JAMBU. *Submission to CAESAR*, 2014.
- [138] Hongjun Wu. The Hash Function JH. Submission to NIST.
- [139] Hongjun Wu. The Stream Cipher HC-128. In *New Stream Cipher Designs*, pages 39–47. Springer, 2008.
- [140] Hongjun Wu, Feng Bao, Robert H. Deng, and Qin-Zhong Ye. Improved truncated differential attacks on safer. In K. Ohta and D. Pei, editors, *ASIACRYPT’98, LNCS 1514*, pages 133–147. Springer-Verlag Berlin Heidelberg, 1998.
- [141] Hongjun Wu and Bart Preneel. Resynchronization Attacks on WG and LEX. In *Fast Software Encryption – FSE 2006*, pages 422–432. Springer, 2006.

- [142] Hongjun Wu and Bart Preneel. *Differential-Linear Attacks Against the Stream Cipher Phelix*, pages 87–100. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.
- [143] Shengbao Wu and Mingsheng Wang. Security Evaluation against Differential Cryptanalysis for Block Cipher Structures. Cryptology ePrint Archive: Report 2011/551, 2011. <http://eprint.iacr.org/>.
- [144] Shengbao Wu, Hongjun Wu, Tao Huang, Mingsheng Wang, and Wenling Wu. Leaked-State-Forgery Attack against the Authenticated Encryption Algorithm ALE. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013*, volume 8269 of *Lecture Notes in Computer Science*, pages 377–404. Springer Berlin Heidelberg, 2013.
- [145] Wenling Wu and Dengguo Feng. *Differential-Linear Cryptanalysis of Camellia*, pages 173–180. Springer US, Boston, MA, 2004.
- [146] Song Y. Yan. *Computational Number Theory and Modern Cryptography*. Higher Education Press Beijing, 2013.
- [147] Z. Yuan, W. Wang, K. Jia, G. Xu, and X. Wang. New Birthday Attacks on Some MACs Based on Block Ciphers. In *Advances in Cryptology – CRYPTO 2009*, pages 209–230. Springer, 2009.
- [148] G. Yuval. How to swindle rabin. *Cryptologia*, 3(3):187–191, 1979.
- [149] Rui Zong, Xiaoyang Dong, and Xiaoyun Wang. Impossible Differential Attack on Simpira v2. Cryptology ePrint Archive, Report 2016/1161, 2016. <http://eprint.iacr.org/2016/1161>.

List of Figures

1.1	Round i of Feistel cipher	20
1.2	Round i of Type-1 Feistel Scheme	21
1.3	Round i of Type-2 Feistel Scheme	22
1.4	Round i of Type-3 Feistel Scheme	22
1.5	Round $2i - 1$ and Round $2i$ of Alternating Feistel Scheme	23
1.6	Round s of Unbalanced Feistel Scheme	24
2.1	Round $3i, 3i + 1, 3i + 2$ of Simpira-3, $0 \leq i \leq 6$	49
2.2	Backward Difference Propagation of the 8-round Simpira-3 $(0, 0, a) \xrightarrow{8R}$ $(F(a), r, r)$	52
2.3	Four possible pattern of a . Filled cells denote arbitrary differences as long as they are not all zero.	53
2.4	Respective patterns of $MC^{-1}(F(a))$	53
2.5	Round 9 in Distinguishing Attack.	54
2.6	Four possible patterns of b	59
2.7	Patterns of $MC^{-1}(F(b))$ corresponding to Figure 2.6.	59
2.8	8-round impossible differential used for 9-round impossible distinguisher	61
3.1	Round i of Type-1 Feistel Scheme	78
3.2	Round i of Type-3 Feistel Scheme	91
3.3	Round $2i - 1$ and Round $2i$ of Alternating Feistel Scheme	94

3.4	Round s of Unbalanced Feistel Scheme	97
4.1	General scheme of ICEPOLE	110
5.1	General scheme of COFFE	139

Appendix A

Appendix

A.1 Session Key Generation Analysis

Recall that here we are trying to investigate the session key generation step whether we can introduce a collision between two valid input such that one has 1– byte value for the domain value, named $\mathcal{S}_1$, and another with 2– bytes value for the domain value, $\mathcal{S}_2$. Here we need to consider the number of bytes for $\mathcal{L}_{K_1}, \mathcal{L}_{V_1}$, and whether zero padding exists in $\mathcal{S}_1$. We will assume that if a string has length that needs to be expressed as a 2-bytes value, $256\mathbf{a} + \mathbf{b}$, then $\mathbf{a} \neq 0$ since otherwise it is not necessary to express the length as a 2-bytes value. We also can assume that for any secret keys considered, it cannot have length 0 since otherwise, there is no need to perform any attack here.

I. **$\mathcal{S}_1$ has no zero padding.** Let $\mathcal{S}_1 = (\mathcal{K}_1 \parallel \mathcal{V}_1 \parallel \mathcal{L}_{K_1} \parallel \mathcal{L}_{V_1} \parallel [\mathbf{0}]_1)$. We will divide this case into smaller sub-cases to handle independently.

I.1. Case I.1. $[\mathcal{L}_{K_1}] = [\mathcal{L}_{V_1}] = 1$. By observation 5.3.2, we have that $\mathcal{L}_{V_1} = 0$.

Let $1 \leq \mathbf{a} \leq 255$ such that $\mathcal{L}_{K_1} = \mathbf{a}$. Then we have:

$$\mathcal{S}_1 = (\mathcal{K}_1 \parallel \mathbf{a} \parallel [0]_2).$$

Now we consider the second string $\mathcal{S}_2$ with the domain value being expressed by 2-bytes string. Then we have $\mathcal{L}_{V_2} \geq \mathbf{a}$. Since we need at least 1 byte to represent $\mathcal{L}_{K_2}$ and a non-empty string of $\mathcal{K}_2$, we need at least $\mathbf{a} + 2$ bits in the remaining unused string in $\mathcal{S}_2$. However, since we want $\mathcal{S}_1 = \mathcal{S}_2$, we only have $\mathbf{a}$ bits left that is unused. Hence if $[\mathcal{L}_{K_1}] = [\mathcal{L}_{V_1}] = 1$, it is impossible for us to have a collision between $\mathcal{S}_1$ and $\mathcal{S}_2$.

I.2. Case I.2. $[\mathcal{L}_{K_1}] = 2$ and $[\mathcal{L}_{V_1}] = 1$. Observation 5.3.2 implies that $\mathcal{L}_{V_1} = 0$. Let $\mathcal{L}_{K_1} = 256\mathbf{a} + \mathbf{b}$ for some $0 \leq \mathbf{a}, \mathbf{b} \leq 255$ where $\mathbf{a} \neq 0$. So we have:

$$\mathcal{S}_1 = (\mathcal{K}_1 \parallel \mathbf{a} \parallel \mathbf{b} \parallel [0]_2).$$

Now we consider the second string $\mathcal{S}_2$. We have different cases depending on the number of bytes for $\mathcal{L}_{K_2}$ and $\mathcal{L}_{V_2}$.

- Case I.2.a. $[\mathcal{L}_{K_2}] = [\mathcal{L}_{V_2}] = 1$. Then we have that $\mathcal{L}_{K_2} = \mathbf{a}$ and $\mathcal{L}_{V_2} = \mathbf{b}$. Now if $\mathcal{S}_2$ has no zero padding, then we have that $256\mathbf{a} + \mathbf{b} = \mathbf{a} + \mathbf{b}$ which implies that $\mathbf{a} = 0$ which contradicts the fact that $\mathcal{L}_{K_1}$ needing to be expressed as a 2-bytes value. So $\mathcal{S}_2$ must have zero padding. A simple comparison tells us that there will be $255\mathbf{a}$ bits of zero paddings in $\mathcal{S}_2$. However, remember that this zero padding comes from $\mathcal{K}_1$ which is not controllable by us the attacker. Hence we can just hope this is true with probability $2^{-255\mathbf{a}}$. Since $\mathbf{a} \geq 1$, this probability is too small to be feasible. So this case can be discarded from the possible $\mathcal{S}_1$ collision with $\mathcal{S}_2$.
- Case I.2.b. $[\mathcal{L}_{K_2}] = 2, [\mathcal{L}_{V_2}] = 1$. For simplicity, let $\mathcal{K}_1 = \mathcal{K}'_1 \parallel \mathbf{c}$ where $\mathbf{c}$ is the last 8 bits of $\mathcal{K}_1$. Then $\mathcal{L}_{K_2} = 256\mathbf{c} + \mathbf{a}$ and $\mathcal{L}_{V_2} = \mathbf{b}$. Let t' be the number of bits of zero padding in $\mathcal{S}_2$. Then equating the length of $\mathcal{S}_1$ and $\mathcal{S}_2$, we get $256\mathbf{a} + \mathbf{b} = 256\mathbf{c} + \mathbf{a} + \mathbf{b} + t' + 8$ where the 8 comes from $\mathbf{c}$ being a part of $\mathcal{K}_1$ despite it not being a part of $\mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'}$.

This can be simplified to $255\mathbf{a} = 256\mathbf{c} + t' + 8$. So we define

$$\mathcal{F}_{(2,1),(2,1)} = \{(\mathbf{a}, \mathbf{b}, \mathbf{c}, t') : 0 \leq \mathbf{a}, \mathbf{b}, \mathbf{c} \leq 255, t' \geq 0, \mathbf{a}, \mathbf{c} \neq 0,$$

$$255\mathbf{a} = 256\mathbf{c} + t' + 8\}.$$

Now we consider the feasibility of each quadruple $(\mathbf{a}, \mathbf{b}, \mathbf{c}, t') \in \mathcal{F}_{(2,1),(2,1)}$. Let $(\mathbf{a}, \mathbf{b}, \mathbf{c}, t') \in \mathcal{F}_{(2,1),(2,1)}$. Since we have $\mathcal{K}'_1 = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'}$, it is clear that $|\mathcal{K}_1| > |\mathcal{K}_2|$. Then we have $256\mathbf{a} + \mathbf{b} - (256\mathbf{c} + \mathbf{a}) = 255\mathbf{a} + \mathbf{b} - 256\mathbf{c}$ bits of $\mathcal{K}_1$ which needs to be known or fixed. Like before, we assume the bits of $\mathcal{K}_1$ which is in $\mathcal{K}_2$ coincide with probability one since we assume that $\mathcal{K}_1$ is the extension of $\mathcal{K}_2$ by appending it with some bits. So the probability of this happening is $2^{-(255\mathbf{a} + \mathbf{b} - 256\mathbf{c})}$. A simple calculation tells us that we can set $255\mathbf{a} + \mathbf{b} - 256\mathbf{c}$ to be as low as 8. Now for any integer $k \geq 8$, we define $\mathcal{F}_{(2,1),(2,1),k} = \{(\mathbf{a}, \mathbf{b}, \mathbf{c}, t') \in \mathcal{F}_{(2,1),(2,1)}, 255\mathbf{a} + \mathbf{b} - 256\mathbf{c} \geq k\}$. Then if we take the quadruple $(\mathbf{a}, \mathbf{b}, \mathbf{c}, t') \in \mathcal{F}_{(2,1),(2,1),k}$, we know that the probability of collision to happen is at least 2^{-k} . We also need to note that since $\mathcal{K}_1 = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'} \parallel \mathbf{c}$, we must have the last $t' + 8$ bits of $\mathcal{K}_1$ to be $0^{t'} \parallel \mathbf{c}$. So any collision we derive from this family, $\mathcal{K}_1$ cannot be chosen to be any keys. The collision will only happen in the subfamily of the $256\mathbf{a} + \mathbf{b}$ -bits string such that the last $t' + 8$ bits must be $0^{t'} \parallel \mathbf{c}$.

- Case I.2.c. $[\mathcal{L}_{V_2}] = 2$. Then $\mathcal{L}_{V_2} = 256\mathbf{a} + \mathbf{b}$. Note that here we again have $\mathcal{K}'_1 = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^t$ where $\mathcal{K}'_1$ is a $256\mathbf{a} + \mathbf{b} - 8$ bits binary string. However, this is impossible since $\mathcal{V}_2$ itself should have $256\mathbf{a} + \mathbf{b}$ bits by itself. So this case is again impossible.

I.3. Case I.3. $[\mathcal{L}_{K_1}] = 1, [\mathcal{L}_{V_1}] = 2$. By observation 5.3.2, $\mathcal{L}_{V_1}$ is a multiple of 256. Let $1 \leq \mathbf{a}, \mathbf{b} \leq 255$ such that $\mathcal{L}_{K_1} = \mathbf{a}$ and $\mathcal{L}_{V_1} = 256\mathbf{b}$. Then we have

$$\mathcal{S}_1 = (\mathcal{K}_1 \parallel \mathcal{V}_1 \parallel \mathbf{a} \parallel \mathbf{b} \parallel [0]_2).$$

Now we consider the possibilities of $\mathcal{S}_2$.

- Case I.3.a $[\mathcal{L}_{K_2}] = [\mathcal{L}_{V_2}] = 1$. Then $\mathcal{L}_{K_2} = \mathbf{a}$ and $\mathcal{L}_{V_2} = \mathbf{b}$. Then certainly $\mathcal{S}_2$ must have zero paddings. More precisely, we must have $\mathcal{K}_1 = \mathcal{K}_2$ and $\mathcal{V}_1 = \mathcal{V}_2 \parallel 0^{255\mathbf{b}}$. So for any choice of key and nonce, we must have the collision with probability one as long as $\mathcal{V}_1 = \mathcal{V}_2 \parallel 0^{255\mathbf{b}}$.
- Case I.3.b $[\mathcal{L}_{K_2}] = 2, [\mathcal{L}_{V_2}] = 1$. Let $\mathcal{V}_1 = \mathcal{V}'_1 \parallel \mathbf{c}$. Then we have $\mathcal{L}_{K_2} = 256\mathbf{c} + \mathbf{a}$ and $\mathcal{L}_{V_2} = \mathbf{b}$. Now let $\mathcal{S}_2$ to have t' -bits zero padding. Then equating the length of $\mathcal{S}_1$ and $\mathcal{S}_2$, we have $\mathcal{K}_1 \parallel \mathcal{V}_1 = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'} \parallel \mathbf{c}$. So we have $\mathbf{a} + 256\mathbf{b} = 256\mathbf{c} + \mathbf{a} + \mathbf{b} + t' + 8$. Then we have $255\mathbf{b} = 256\mathbf{c} + 8 + t'$. As before, we define a family

$$\mathcal{F}_{(1,2),(2,1)} = \{(\mathbf{a}, \mathbf{b}, \mathbf{c}, t') : 1 \leq \mathbf{a}, \mathbf{b}, \mathbf{c} \leq 255, t' \geq 0, 255\mathbf{b} = 256\mathbf{c} + 8 + t'\}.$$

Now we consider the feasibility of each quadruple $(\mathbf{a}, \mathbf{b}, \mathbf{c}, t') \in \mathcal{F}_{(1,2),(2,1)}$. Let $(\mathbf{a}, \mathbf{b}, \mathbf{c}, t') \in \mathcal{F}_{(1,2),(2,1)}$. Then the length of the bits that is in $\mathcal{K}_2$ but not in $\mathcal{K}_1$ is $256\mathbf{c}$. Now remember that since $\mathcal{L}_{K_2} = 256\mathbf{c} + \mathbf{a}$, $\mathbf{c} \neq 0$. So the probability of collision to happen is at most 2^{-256} which is infeasible. So we can disregard this case.

- Case I.3.c $[\mathcal{L}_{K_2}] = 1, [\mathcal{L}_{V_2}] = 2$. This case has been discussed in detail in the main section. So we will skip the discussion of this case here.
- Case I.3.d $[\mathcal{L}_{K_2}] = [\mathcal{L}_{V_2}] = 2$. Let $\mathcal{V}_1 = \mathcal{V}''_1 \parallel \mathbf{d} \parallel \mathbf{c}$ where $\mathbf{d} \parallel \mathbf{c}$ is the last 2-bytes of $\mathcal{V}_1$. So $\mathcal{L}_{K_2} = 256\mathbf{d} + \mathbf{c}$ and $\mathcal{L}_{V_2} = 256\mathbf{a} + \mathbf{b}$. Similar as before, we let $\mathcal{S}_2$ to have t' -bits of zero padding. Now equating the length of $\mathcal{S}_1$ and $\mathcal{S}_2$, we have $\mathcal{K}_1 \parallel \mathcal{V}_1 = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'} \parallel \mathcal{L}_{K_2}$. In other words, $\mathbf{a} + 256\mathbf{b} = 256\mathbf{d} + \mathbf{c} + 256\mathbf{a} + \mathbf{b} + t' + 16$ or equivalently

$$255(\mathbf{b} - \mathbf{a}) = 256\mathbf{d} + \mathbf{c} + t' + 16.$$

Let $\mathcal{F}_{(1,2),(2,2)}$ be the family:

$$\mathcal{F}_{(1,2),(2,2)} = \{(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, t') : 1 \leq \mathbf{a}, \mathbf{b}, \mathbf{d} \leq 255, 0 \leq \mathbf{c} \leq 255, t' \geq 0,$$

$$255(\mathbf{b} - \mathbf{a}) = 256\mathbf{d} + \mathbf{c} + t' + 16\}.$$

Now we investigate each quintuple in this family. Let $(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, t') \in \mathcal{F}_{(1,2),(2,2)}$. Obviously, since $[\mathcal{K}_2] = 2$ and $[\mathcal{K}_1] = 1$, $|\mathcal{K}_2| > |\mathcal{K}_1|$. Note that the number of bits that is in $\mathcal{K}_2$ but not in $\mathcal{K}_1$, and hence needs to be guessed, is $256\mathbf{d} + \mathbf{c} - \mathbf{a}$. This gives us the success probability of the collision to be $2^{\mathbf{a}-256\mathbf{d}-\mathbf{c}}$. A simple calculation tells us that this can be as small as 3, for example when $\mathbf{a} = 253, \mathbf{b} = 255, \mathbf{c} = 0, \mathbf{d} = 1, t' = 238$. Now as before, for any integer $k \geq 3$, we define a subclass $\mathcal{F}_{(1,2),(2,2),k}$ of $\mathcal{F}_{(1,2),(2,2)}$ such that $\mathcal{F}_{(1,2),(2,2),k} = \{(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, t) \in \mathcal{F}_{(1,2),(2,2)} : 256\mathbf{d} + \mathbf{c} - \mathbf{a} \geq k\}$. Note that here since $\mathcal{K}_2$ is the longer key and $\mathcal{K}_1 \parallel \mathcal{V}_1'' = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'} \parallel \mathcal{L}_{K_2}$, the extension of the $\mathcal{K}_2$ but not in $\mathcal{K}_1$ must be in $\mathcal{V}_1''$ which is controllable. So any choice of $\mathcal{K}_2$ is vulnerable to the collision.

I.4. Case I.4. $[\mathcal{L}_{K_1}] = [\mathcal{L}_{V_1}] = 2$. By observation 5.3.2, we have $\mathcal{L}_{V_1}$ is a multiple of 256. Let $1 \leq \mathbf{a}, \mathbf{c} \leq 255, 0 \leq \mathbf{b} \leq 255$ such that $\mathcal{L}_{K_1} = 256\mathbf{a} + \mathbf{b}$ and $\mathcal{L}_{V_1} = 256\mathbf{c}$. Then we have:

$$\mathcal{S}_1 = (\mathcal{K}_1 \parallel \mathcal{V}_1 \parallel \mathbf{a} \parallel \mathbf{b} \parallel \mathbf{c} \parallel [0]_2).$$

Dividing the case further based on $\mathcal{S}_2$, we have the following cases:

- Case I.4.a $[\mathcal{L}_{K_2}] = [\mathcal{L}_{V_2}] = 1$. Then $\mathcal{L}_{K_2} = \mathbf{b}$ and $\mathcal{L}_{V_2} = \mathbf{c}$. Now let $\mathcal{S}_2$ to have t' bits of zero padding. Now equating the length of $\mathcal{S}_1$ and $\mathcal{S}_2$, we have $\mathcal{K}_1 \parallel \mathcal{V}_1 \parallel \mathbf{a} = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'}$. So we have $256\mathbf{a} + \mathbf{b} + 256\mathbf{c} + 8 = \mathbf{b} + \mathbf{c} + t'$. So we have $t' = 256\mathbf{a} + 255\mathbf{c} + 8$. Now considering the string $\mathcal{K}_1 \parallel \mathcal{V}_1 \parallel \mathbf{a} = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'}$ from the right, we have that since $\mathbf{a} \neq 0$, we must have $t' \leq 7$. Since otherwise, this will force $\mathbf{a} = 0$. So we have $256\mathbf{a} + 255\mathbf{c} + 8 \leq 7$. However, this is clearly impossible since $1 \leq \mathbf{a}, \mathbf{c}$. So we can disregard this case.
- Case I.4.b $[\mathcal{L}_{K_2}] = 2, [\mathcal{L}_{V_2}] = 1$. Then $\mathcal{L}_{K_2} = 256\mathbf{a} + \mathbf{b}$ and $\mathcal{L}_{V_2} = \mathbf{c}$. Letting $\mathcal{S}_2$ to have t -bits of zero padding, equating $\mathcal{S}_1$ to $\mathcal{S}_2$, we have

$\mathcal{K}_1 \parallel \mathcal{V}_1 = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^t$. Now since $\mathcal{L}_{K_1} = \mathcal{L}_{K_2}$, we must have $\mathcal{K}_1 = \mathcal{K}_2$. So this leaves us with $\mathcal{V}_1 = \mathcal{V}_2 \parallel 0^t$ which implies $t = 255\mathbf{c}$.

- Case I.4.c $[\mathcal{L}_{K_2}] = 1, [\mathcal{L}_{V_2}] = 2$. Then we have $\mathcal{L}_{K_2} = \mathbf{a}$ and $\mathcal{L}_{V_2} = 256\mathbf{b} + \mathbf{c}$. Now inspecting the first part of both $\mathcal{S}_1$ and $\mathcal{S}_2$, we must have at least $255\mathbf{a} + \mathbf{b}$ bits of $\mathcal{K}_1$ that is not in $\mathcal{K}_2$ which means it must be guessed. Now since $\mathbf{a} \neq 0$, we have the probability of success to be at most 2^{-255} which is already infeasible. So we can disregard this case.
- Case I.4.d $[\mathcal{L}_{K_2}] = [\mathcal{L}_{V_2}] = 2$. For this, we let $\mathcal{V}_1 = \mathcal{V}'_1 \parallel \mathbf{d}$ where $\mathbf{d}$ is the last byte of $\mathcal{V}_1$. Then we have $\mathcal{L}_{K_2} = 256\mathbf{d} + \mathbf{a}$ and $\mathcal{L}_{V_2} = 256\mathbf{b} + \mathbf{c}$. Let $\mathcal{S}_2$ have t -bits of zero padding. Then equating $\mathcal{S}_1$ to $\mathcal{S}_2$, we have $\mathcal{K}_1 \parallel \mathcal{V}'_1 = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'}$. So we have $256\mathbf{a} + \mathbf{b} + 256\mathbf{c} - 8 = 256\mathbf{d} + \mathbf{a} + 256\mathbf{b} + \mathbf{c} + t'$. Equivalently, we have

$$255(\mathbf{a} - \mathbf{b} + \mathbf{c}) = 256\mathbf{d} + t' + 8.$$

Let $\mathcal{F}_{(2,2),(2,2)}$ be the family:

$$\mathcal{F}_{(2,2),(2,2)} = \{(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, t') : 1 \leq \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \leq 255, t' \geq 0,$$

$$255(\mathbf{a} - \mathbf{b} + \mathbf{c}) = 256\mathbf{d} + t' + 8\}.$$

Now we investigate the feasibility of each quintuple in the family. Let $(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, t') \in \mathcal{F}_{(2,2),(2,2)}$. Note that there are $|256\mathbf{a} + \mathbf{b} - (256\mathbf{d} + \mathbf{a})| = |255\mathbf{a} + \mathbf{b} - 256\mathbf{d}|$ bits that is in one but not both secret keys. This means that the probability of success is at most $2^{-|255\mathbf{a} + \mathbf{b} - 256\mathbf{d}|}$. We can perform a simple calculation to discover that the value $|255\mathbf{a} + \mathbf{b} - 256\mathbf{d}|$ can be made to be 0, for example, when $(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, t') = (1, 1, 2, 1, 246)$. For this purpose, for any non-negative integers k , we define a subclass $\mathcal{F}_{(2,2),(2,2),k}$ of $\mathcal{F}_{(2,2),(2,2)}$ such that $\mathcal{F}_{(2,2),(2,2),k} = \{(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, t) \in \mathcal{F}_{(2,2),(2,2)} : |255\mathbf{a} + \mathbf{b} - 256\mathbf{d}| \leq k\}$. So for any quintuple extracted

from $\mathcal{F}_{(2,2),(2,2),k}$, it can be used to guess the extension of at most k bits of the secret key with the success probability being at least 2^{-k} . Now we need to investigate the extension of the secret key. Recall that $\mathcal{K}_1 \parallel \mathcal{V}'_1 = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'}$. Now if $\mathcal{K}_2$ is longer, then the extension will all be in $\mathcal{V}'_1$ considering the left term only has $\mathcal{V}'_1$ in addition to $\mathcal{K}_1$. So for this case, there is no restriction for either key. Now we check the case when $\mathcal{K}_1$ is longer. Note that if $\mathcal{K}_1$ is longer, then the extension is from $\mathcal{V}_2 \parallel 0^{t'}$. Now if $\mathcal{V}'_1$ is longer than t' , then since the two strings are equal, no bits from the extension of $\mathcal{K}_1$ can be from the zero padding. So if $t' < \mathcal{L}_{V'_1} = \mathcal{L}_{V_1} - 8$, then there are no restriction to $\mathcal{K}_1$. On the other hand, if $t' > \mathcal{L}_{V_1} - 8$, then the last $t' - (\mathcal{L}_{V_1} - 8)$ bits of $\mathcal{K}_1$ must be zero. So in other words, when $\mathcal{K}_1$ is longer, the last $\max(0, t' - (\mathcal{L}_{V_1} - 8))$ bits of it must be 0.

II. $\mathcal{S}_1$ has a t -bits zero padding. So now we have $\mathcal{S}_1 = (\mathcal{K}_1 \parallel \mathcal{V}_1 \parallel 0^t \parallel \mathcal{L}_{K_1} \parallel \mathcal{L}_{V_1} \parallel [0]_1)$.

We will again consider some small sub-cases to analyse.

II.1. Case II.1. $[\mathcal{L}_{K_1}] = [\mathcal{L}_{V_1}] = 1$ As before, Observation 5.3.2 tells us that $\mathcal{L}_{V_1} = 0$. Let $1 \leq \mathbf{a} \leq 255$ such that $\mathcal{L}_{K_1} = \mathbf{a}$. Then we have:

$$\mathcal{S}_1 = (\mathcal{K}_1 \parallel 0^t \parallel \mathbf{a} \parallel [0]_2).$$

Now we consider the second string $\mathcal{S}_2$. Note that here $\mathcal{L}_{V_2} \geq \mathbf{a}$. So we will divide this to two cases:

- Case II.1.a. $\mathcal{L}_{V_2} = \mathbf{a}$. We also have that $\mathcal{K}_1 \parallel 0^t = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'} \parallel \mathcal{L}_{K_2}$ where t' is the length of the zero padding of $\mathcal{S}_2$. Now since we need the next 1 or 2 bytes in the left of $\mathcal{L}_{V_2}$ to be $\mathcal{L}_{K_2}$ and it cannot be 0, it implies that $t < 8$ if $[\mathcal{L}_{K_2}] = 1$ and $t < 16$ if $[\mathcal{L}_{K_2}] = 2$. Now note that here $\mathcal{K}_1$ has the same length as $\mathcal{V}_2$. So for the two strings to be the same, we need the sum of the length of $\mathcal{K}_2, 0^{t'}$, and $\mathcal{L}_{K_2}$ must be

equal to t , which must be strictly less than 16. This tells us that $\mathcal{L}_{K_2}$ cannot be a 2-bytes value.

So this gives us that $[\mathcal{L}_{K_2}] = 1$ and $t < 8$. Let $\mathbf{b}$ be the last $8-t$ bits of $\mathcal{K}_1$ which must satisfy $\mathbf{b} \neq 0$. Then we have $\mathcal{L}_{K_2} = \mathbf{b}.2^t$. So equating the length here, we have that $\mathbf{a} + t = \mathbf{b}.2^t + \mathbf{a} + t' + 8$ or equivalently, $(t-8) - \mathbf{b}.2^t = t - \mathbf{b}.2^t - 8 = t' \geq 0$. Now remember that $t-8 < 0$ and $-\mathbf{b}.2^t < 0$. The sum of two negative numbers cannot be non-negative. So this case can be discarded.

- Case II.1.b. $\mathcal{L}_{V_2} > \mathbf{a}$. So $[\mathcal{L}_{V_2}] = 2$. Now if $t > 8$, we will have that the most significant byte of $\mathcal{L}_{V_2}$ is zero, which cannot be by the observation we made in the very beginning of this section. So we have $t < 8$. We again let $\mathbf{b}$ to be the last $8-t$ bits of $\mathcal{K}_1$ and $\mathbf{b} \neq 0$. Then we have $\mathcal{L}_{V_2} = \mathbf{b}.2^{t+8} + \mathbf{a}$ and the remaining string to be $\mathcal{K}'_1$. Assuming $\mathcal{S}_2$ has t' -bits of zero padding, equating the strings, we have $\mathcal{K}'_1 = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'} \parallel \mathcal{L}_{K_1}$. Recall that here $\mathcal{K}'_1$ must have length strictly less than 256 bits since $[\mathcal{L}_{K_1}] = 1$ and it has been truncated by $8-t$ bits. However, by assumption, we must already have $\mathcal{V}_2$ itself to have length more than 256 bits. Obviously, this is impossible. So we can disregard this case.

II.2. Case II.2. $[\mathcal{L}_{K_1}] = 2, [\mathcal{L}_{V_1}] = 1$. By Observation 5.3.2, we have $\mathcal{L}_{V_1} = 0$. Let $0 \leq \mathbf{a}, \mathbf{b} \leq 255, \mathbf{a} \neq 0$ such that $\mathcal{L}_{K_1} = 256\mathbf{a} + \mathbf{b}$. Then we have

$$\mathcal{S}_1 = (\mathcal{K}_1 \parallel 0^t \parallel \mathbf{a} \parallel \mathbf{b} \parallel [0]_2).$$

Here, considering $\mathcal{S}_2$, we again have two cases:

- Case II.2.a $\mathcal{L}_{V_2} = 256\mathbf{a} + \mathbf{b}$. Then equating the length of the two strings, we must have $\mathcal{K}_1 \parallel 0^t = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'} \parallel \mathcal{L}_{K_2}$ where t' is the length of zero padding of $\mathcal{S}_2$. Now since the length of $\mathcal{K}_1$ and $\mathcal{V}_2$ are the same, we must have the sum of the lengths of $\mathcal{K}_2, 0^{t'}$, and $\mathcal{L}_{K_2}$ to

be equal to t . Now note that by the same argument as before, t must be less than 8 if $[\mathcal{L}_{K_2}] = 1$ and $t < 16$ if $[\mathcal{L}_{K_2}] = 2$. Now if $\mathcal{L}_{K_2}$ is a 2-byte string, the length of the string in the right hand side must exceed 256 bits. However, the one on the left cannot even exceed 16. So we cannot have $\mathcal{L}_{K_2}$ to be a 2-byte string. So we have $[\mathcal{L}_{K_2}] = 1$. So we have $t < 8$. Furthermore, we have that $t = \mathcal{L}_{K_2} + t' + 8$ where the addition by 8 comes from the fact that $\mathcal{L}_{K_2}$ is an 8-bit string. However, this is impossible since all the terms here are non-negative and this causes the left hand side to be strictly less than 8 while the right hand side is at least 8. So we can again disregard this case.

- Case II.2.b. $\mathcal{L}_{V_2} = \mathbf{b}$. For this, we further can divide the case into two depending on $[\mathcal{L}_{K_2}]$. First of all, if $\mathcal{L}_{K_2}$ is a 1-byte value, then we have $\mathcal{L}_{K_2} = \mathbf{a}$. From here we see that there must be $255\mathbf{a} + \mathbf{b}$ bits that can be found in $\mathcal{K}_1$ but not in $\mathcal{K}_2$. Now since $\mathbf{a} \neq 0$, this means at least 255-bits of secret string that we need to guess. This implies that the success probability of this case is at most 2^{-255} which is already infeasible. So we need $[\mathcal{L}_{K_2}] = 2$. By the same observation as before, $t < 8$. Let $\mathcal{K}_1 = \mathcal{K}'_1 \parallel \mathbf{c}$ where $\mathbf{c}$ is the last $8 - t$ bits of $\mathcal{K}_1$. Then we have that $\mathcal{L}_{K_2} = \mathbf{c}.2^{8+t} + \mathbf{a}$. Now we let $\mathcal{S}_2$ have t' -bits of zero padding. Then we have $\mathcal{K}'_1 = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'}$. So equating the length, we have that $256\mathbf{a} + \mathbf{b} - (8 - t) = \mathbf{c}.2^{8+t} + \mathbf{a} + \mathbf{b} + t'$. Equivalently, we have that

$$255\mathbf{a} = \mathbf{c}.2^{8+t} + t' + 8 - t.$$

Note that this also tells us that $\mathcal{L}_{K_1} \geq \mathcal{L}_{K_2}$. As before, we define a family:

$$\mathcal{F}_{\mathbf{p},(2,1),(2,1)} = \{(\mathbf{a}, \mathbf{b}, \mathbf{c}, t, t') : 1 \leq \mathbf{a}, \mathbf{b} \leq 255, 1 \leq t < 8, 1 \leq \mathbf{c} \leq 2^{8-t} - 1$$

$$t' \geq 0, 255\mathbf{a} = \mathbf{c}.2^{8+t} + t' + 8 - t\}.$$

Now we consider the feasibility of each quadruple in the family.

Let $(\mathbf{a}, \mathbf{b}, \mathbf{c}, t, t') \in \mathcal{F}_{\mathbf{p},(2,1),(2,1)}$. Then the length of the bits in $\mathcal{K}_1$ but not in $\mathcal{K}_2$ is $256\mathbf{a} + \mathbf{b} - (\mathbf{c}.2^{8+t} + \mathbf{a}) = 255\mathbf{a} + \mathbf{b} - (\mathbf{c}.2^{8+t})$. Now a simple enumeration of the cases tells us that this value can be kept as low as 8. An example of such quintuple is $(\mathbf{a}, \mathbf{b}, \mathbf{c}, t, t') = (249, 1, 124, 1, 14)$. One more thing to note here is that since $\mathcal{K}_1 = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'} \parallel \mathbf{c}$, we have that the last $t' + 8 - t$ bits of $\mathcal{K}_1$ is fixed to be $0^{t'} \parallel \mathbf{c}$. So here we have a restriction of the $\mathcal{K}_1$ that is vulnerable to the collision. The secret string must have its last $t' + 8 - t$ bits to be $0^{t'} \parallel \mathbf{c}$. For any integers $k \geq 8$, we define a subfamily $\mathcal{F}_{\mathbf{p},(2,1),(2,1),k}$ of $\mathcal{F}_{\mathbf{p},(2,1),(2,1)}$ satisfying:

$$\mathcal{F}_{\mathbf{p},(2,1),(2,1),k} = \{(\mathbf{a}, \mathbf{b}, \mathbf{c}, t, t') \in \mathcal{F}_{\mathbf{p},(2,1),(2,1)} : 255\mathbf{a} + \mathbf{b} - (\mathbf{c}.2^{8+t}) \leq k\}.$$

Taking any quintuple of $\mathcal{F}_{\mathbf{p},(2,1),(2,1),k}$, we can use this to guess at most k -bits of key extension with success probability at least 2^{-k} .

II.3. Case II.3. $[\mathcal{L}_{K_1}] = 1, [\mathcal{L}_{V_1}] = 2$. By observation 5.3.2, we must have $\mathcal{L}_{V_1}$ to be divisible by 256. So we let $1 \leq \mathbf{a}, \mathbf{b} \leq 255$ such that $\mathcal{L}_{K_1} = \mathbf{a}$ and $\mathcal{L}_{V_1} = 256\mathbf{b}$. Then we have:

$$\mathcal{S}_1 = (\mathcal{K}_1 \parallel \mathcal{V}_1 \parallel 0^t \parallel \mathbf{a} \parallel \mathbf{b} \parallel [0]_2).$$

Now as before, we consider the number of bytes of $\mathcal{L}_{K_2}$ and $\mathcal{L}_{V_2}$.

- Case II.3.a. $[\mathcal{L}_{K_2}] = [\mathcal{L}_{V_2}] = 1$. Then $\mathcal{L}_{K_2} = \mathbf{a}$ and $\mathcal{L}_{V_2} = \mathbf{b}$. Now $\mathcal{L}_{K_1} = \mathcal{L}_{K_2}$ and they are the first $\mathbf{a}$ bits of the strings $\mathcal{S}_1 = \mathcal{S}_2$. So $\mathcal{K}_1 = \mathcal{K}_2$. This implies that $\mathcal{V}_2$ should be the first $\mathbf{b}$ bits of $\mathcal{V}_1$ and the next $255\mathbf{b}$ bits of $\mathcal{V}_1$ should be the zero padding so that it can be absorbed by the zero padding of $\mathcal{S}_2$.
- Case II.3.b. $[\mathcal{L}_{K_2}] = 2, [\mathcal{L}_{V_2}] = 1$. Using the same analysis as before, we have $t < 8$ and we can let $\mathcal{V}_1 = \mathcal{V}'_1 \parallel \mathbf{c}$ where $\mathbf{c}$ is the last $8 - t$

bits of $\mathcal{V}_1$. Then we have $\mathcal{L}_{K_2} = \mathbf{c}.2^{8+t} + \mathbf{a}$ and $\mathcal{L}_{V_2} = \mathbf{b}$. Now this means that we have $\mathbf{c}.2^{8+t}$ bits that is in $\mathcal{K}_2$ but not in $\mathcal{K}_1$. Now since we have $t > 0$ and $\mathbf{c} \neq 0$, the success probability of the collision is at most 2^{-256} which is infeasible. So we can disregard this case.

- Case II.3.c. $[\mathcal{L}_{K_2}] = 1, [\mathcal{L}_{V_2}] = 2$. By the same analysis as before, we get that $t < 8$ and we can let $\mathcal{V}_1 = \mathcal{V}'_1 \parallel \mathbf{c}$ where $\mathbf{c}$ is the last $8 - t$ bits of $\mathcal{V}_1$. So we have $\mathcal{L}_{V_2} = 256\mathbf{a} + \mathbf{b}$ and $\mathcal{L}_{K_2} = \mathbf{c}.2^t$. Assuming $\mathcal{S}_2$ to have t' bits of zero padding, we have $\mathcal{K}_1 \parallel \mathcal{V}'_1 = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'}$. So we have $\mathbf{a} + 256\mathbf{b} - (8 - t) = \mathbf{c}.2^t + 256\mathbf{a} + \mathbf{b} + t'$. Simplifying this, we get:

$$255(\mathbf{b} - \mathbf{a}) = \mathbf{c}.2^t + 8 - t + t'.$$

So we can consider the family:

$$\mathcal{F}_{\mathbf{p},(1,2),(1,2)} = \{(\mathbf{a}, \mathbf{b}, \mathbf{c}, t, t') : 1 \leq \mathbf{a}, \mathbf{b} \leq 255, 0 < t < 8, 1 \leq \mathbf{c} \leq 2^{(8-t)} - 1,$$

$$t' \geq 0,$$

$$255(\mathbf{b} - \mathbf{a}) = \mathbf{c}.2^t + 8 - t + t'\}.$$

Now we discuss the feasibility of each quintuple in this family. Let $(\mathbf{a}, \mathbf{b}, \mathbf{c}, t, t') \in \mathcal{F}_{\mathbf{p},(1,2),(1,2)}$. So there will be $|\mathbf{a} - \mathbf{c}.2^t|$ -bits of secret string that is in one but not in both of the secret keys. In other words, we will need to guess these bits. A simple calculation reveals that this value can be kept as low as 0. A really small example would be to set $(\mathbf{a}, \mathbf{b}, \mathbf{c}, t, t') = (2, 3, 1, 1, 260)$. For any non-negative integer k , we then can define a non-empty subfamily $\mathcal{F}_{\mathbf{p},(1,2),(1,2),k}$ such that:

$$\mathcal{F}_{\mathbf{p},(1,2),(1,2),k} = \{(\mathbf{a}, \mathbf{b}, \mathbf{c}, t, t') \in \mathcal{F}_{\mathbf{p},(1,2),(1,2)} : |\mathbf{a} - \mathbf{c}.2^t| \leq k\}.$$

Having these subfamilies, we know that if we take a quintuple from $\mathcal{F}_{\mathbf{p},(1,2),(1,2),k}$, then it can help us in guessing the extension of at most k -bits of secret key with success probability at least 2^{-k} . Now we also

still need to investigate the extension secret string. We again equate the two strings to get $\mathcal{K}_1 \parallel \mathcal{V}'_1 = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'}$. Now if $\mathcal{K}_2$ is the longer one, since the left term contains only $\mathcal{K}_1$ and $\mathcal{V}'_1$, the extension can only come from $\mathcal{V}'_1$. So if $\mathcal{K}_2$ is longer, we have no restriction on either $\mathcal{K}_1$ or $\mathcal{K}_2$. On the other hand, if $\mathcal{K}_1$ is the longer one, the same argument cannot be used since the right term contains $0^{t'}$. However, we again notice that $[\mathcal{L}_{K_1}] = [\mathcal{L}_{K_2}] = 1$. So $[\mathcal{L}_{K_1} - \mathcal{L}_{K_2}] = 1$. Now since $[\mathcal{L}_{V_2}] = 2$, certainly $\mathcal{L}_{K_1} - \mathcal{L}_{K_2} < \mathcal{L}_{V_2}$. So again, all the extension can only come from the controllable nonce.

So we proved that in either case, we have no restriction on the choice of the secret key. So any choice of key following the parameters in $\mathcal{F}_{\mathbf{p},(1,2),(1,2)}$ is vulnerable to the collision.

- Case II.3.d. $[\mathcal{L}_{K_2}] = [\mathcal{L}_{V_2}] = 2$. Then we have that $\mathcal{L}_{V_2} = 256\mathbf{a} + \mathbf{b}$. Furthermore, by the same analysis as before, we have $t < 16$ and we can let $\mathcal{V}_1 = \mathcal{V}'_1 \parallel \mathbf{c}$ where $\mathbf{c}$ is the last $16 - t$ -bits of $\mathcal{V}_1$. We also have that if $t < 8$, we need $\mathbf{c} \geq 2^{8-t}$ to ensure $\mathcal{L}_{K_2}$ to be a 2-bytes value. Then we have $\mathcal{L}_{K_2} = \mathbf{c} \cdot 2^t$. Note that this gives us that $\mathcal{L}_{K_1} = \mathbf{a} < 256 \leq \mathbf{c} \cdot 2^t = \mathcal{L}_{K_2}$. So $\mathcal{L}_{K_1} < \mathcal{L}_{K_2}$. Now we have $\mathcal{K}_1 \parallel \mathcal{V}'_1 = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'}$ assuming $\mathcal{S}_2$ has t' -bits of zero padding. So equating the length of these two strings, we have $\mathbf{a} + 256\mathbf{b} - (16 - t) = \mathbf{c} \cdot 2^t + 256\mathbf{a} + \mathbf{b} + t'$ or equivalently

$$255(\mathbf{b} - \mathbf{a}) = \mathbf{c} \cdot 2^t + 16 - t + t'.$$

Now as usual, consider the family

$$\mathcal{F}_{\mathbf{p},(1,2),(2,2)} = \{(\mathbf{a}, \mathbf{b}, \mathbf{c}, t, t') : 1 \leq \mathbf{a}, \mathbf{b} \leq 255, 1 \leq t \leq 15,$$

$$\max(1, 2^{8-t}) \leq \mathbf{c} \leq 2^{16-t}, t' \geq 0, 255(\mathbf{b} - \mathbf{a}) = \mathbf{c} \cdot 2^t + 16 - t + t'\}.$$

Now we consider the feasibility of each quintuple. Let $(\mathbf{a}, \mathbf{b}, \mathbf{c}, t, t') \in \mathcal{F}_{\mathbf{p},(1,2),(2,2)}$. Then we have $\mathbf{c} \cdot 2^t - \mathbf{a}$ bits that is in $\mathcal{K}_2$ that is not in $\mathcal{K}_1$

and hence needs to be guessed. A simple calculation tells us that this value can reach as low as 3, which is achievable, for instance, when $(\mathbf{a}, \mathbf{b}, \mathbf{c}, t, t') = (253, 255, 128, 1, 269)$. So like before, we can define a subfamily of $\mathcal{F}_{\mathbf{p},(1,2),(2,2),k}$ for any integer $k \geq 3$:

$$\mathcal{F}_{\mathbf{p},(1,2),(2,2),k} = \{(\mathbf{a}, \mathbf{b}, \mathbf{c}, t, t') \in \mathcal{F}_{\mathbf{p},(1,2),(2,2)}, \mathbf{c} \cdot 2^t - \mathbf{a} \leq k\}.$$

Then we have that for any quintuple taken from $\mathcal{F}_{\mathbf{p},(1,2),(2,2),k}$, the success probability of collision of the session key is at least 2^{-k} .

II.4. Case II.4. $[\mathcal{L}_{K_1}] = [\mathcal{L}_{V_1}] = 2$. Then we can have $0 \leq \mathbf{a}, \mathbf{b}, \mathbf{c} \leq 255, \mathbf{a}, \mathbf{c} \neq 0$ such that $\mathcal{L}_{K_1} = 256\mathbf{a} + \mathbf{b}$ and $\mathcal{L}_{V_1} = 256\mathbf{c}$. Here 256 divides $\mathcal{L}_{V_1}$ due to Observation 5.3.2. So we have:

$$\mathcal{S}_1 = (\mathcal{K}_1 \parallel \mathcal{V}_1 \parallel 0^t \parallel \mathbf{a} \parallel \mathbf{b} \parallel \mathbf{c} \parallel [0]_2).$$

As before, we further divide this case to 4 smaller subcases based on $[\mathcal{L}_{K_2}]$ and $[\mathcal{L}_{V_2}]$.

- Case II.4.a. $[\mathcal{L}_{K_2}] = [\mathcal{L}_{V_2}] = 1$. Then we have $\mathcal{L}_{K_2} = \mathbf{b}$ and $\mathcal{L}_{V_2} = \mathbf{c}$. Now since $\mathbf{a} \neq 0$, if t' is the number of zero paddings in $\mathcal{S}_2$, we must have $t' < 8$. Then we have $\mathcal{K}_1 \parallel \mathcal{V}_1 \parallel 0^t \parallel \mathbf{a} = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'}$. So equating the length, we have $256\mathbf{a} + \mathbf{b} + 256\mathbf{c} + t + 8 = \mathbf{b} + \mathbf{c} + t'$ or equivalently

$$256\mathbf{a} + 255\mathbf{c} + t = t' - 8 < 0.$$

Now note that $\mathbf{a}, \mathbf{c}, t > 0$. So the expression $256\mathbf{a} + 255\mathbf{c} + t$ must be a positive number. So the inequality cannot be satisfied for any value of the variables. Hence this case is impossible and can be disregarded.

- Case II.4.b. $[\mathcal{L}_{K_2}] = 2$ and $[\mathcal{L}_{V_2}] = 1$. Then we have $\mathcal{L}_{V_2} = \mathbf{c}$ and $\mathcal{L}_{K_2} = 256\mathbf{a} + \mathbf{b}$. As before, this implies that $\mathcal{K}_1 = \mathcal{K}_2$ and $\mathcal{V}_1 = \mathcal{V}_2 \parallel 0^{255\mathbf{c}}$.

- Case II.4.c. $[\mathcal{L}_{K_2}] = 1$ and $[\mathcal{L}_{V_2}] = 2$. Then we have $\mathcal{L}_{V_2} = 256\mathbf{b} + \mathbf{c}$ and $\mathcal{L}_{K_2} = \mathbf{a}$. So we have $255\mathbf{a} + \mathbf{b}$ bits of $\mathcal{K}_1$ that is not in $\mathcal{K}_2$ and hence needs to be guessed. Since $\mathbf{a} \neq 0$, this translates to a success probability of at most 2^{-255} which is infeasible. So we can again disregard this case.
- Case II.4.d. $[\mathcal{L}_{K_2}] = [\mathcal{L}_{V_2}] = 2$. Then we have $\mathcal{L}_{V_2} = 256\mathbf{b} + \mathbf{c}$ and by similar analysis as before, $t < 8$. Furthermore, if we let $\mathcal{V}_1 = \mathcal{V}'_1 \parallel \mathbf{d}$ where $\mathbf{d}$ is the last $8 - t$ -bits of $\mathcal{V}_1$, we have $\mathcal{L}_{K_2} = 2^{8+t}\mathbf{d} + \mathbf{a}$. Then we have $\mathcal{K}_1 \parallel \mathcal{V}'_1 = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'}$ where t' is the number of zero paddings in $\mathcal{S}_2$. So equating the length, we get $256\mathbf{a} + \mathbf{b} + 256\mathbf{c} - (8 - t) = \mathbf{d} \cdot 2^{8+t} + \mathbf{a} + 256\mathbf{b} + \mathbf{c} + t'$. Simplifying this, we have the equation:

$$255(\mathbf{a} - \mathbf{b} + \mathbf{c}) = \mathbf{d} \cdot 2^{8+t} + 8 - t + t'.$$

So we can define the family

$$\mathcal{F}_{\mathbf{p},(2,2),(2,2)} = \{(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, t, t') : 1 \leq \mathbf{a}, \mathbf{b}, \mathbf{c} \leq 255, 1 \leq t \leq 7, 1 \leq \mathbf{d} \leq 2^{8-t}-1, \\ t' \geq 0, 255(\mathbf{a} - \mathbf{b} + \mathbf{c}) = \mathbf{d} \cdot 2^{8+t} + 8 - t + t'\}$$

Next we check the feasibility of each elements of this family. Let $(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, t, t') \in \mathcal{F}_{\mathbf{p},(2,2),(2,2)}$. Then we have $|256\mathbf{a} + \mathbf{b} - (\mathbf{d} \cdot 2^{8+t} + \mathbf{a})| = |255\mathbf{a} + \mathbf{b} - \mathbf{d} \cdot 2^{8+t}|$ secret bits that is in one secret key but not both. So this needs to be guessed. Now enumerating all the elements of the family, we see that this value can be as low as 0 which, for instance, can be realised by the element $(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, t, t') = (2, 2, 3, 1, 1, 260)$. Then as before, we can define a subfamily $\mathcal{F}_{\mathbf{p},(2,2),(2,2),k}$ for any non-negative integers such that

$$\mathcal{F}_{\mathbf{p},(2,2),(2,2),k} = \{(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, t, t') \in \mathcal{F}_{\mathbf{p},(2,2),(2,2)} : |255\mathbf{a} + \mathbf{b} - \mathbf{d} \cdot 2^{8+t}| \leq k\}.$$

Then taking any element from $\mathcal{F}_{\mathbf{p},(2,2),(2,2),k}$, the success probability of a collision is at least 2^{-k} . Lastly, we still need to investigate the extension

of the secret key. Recall that $\mathcal{K}_1 \parallel \mathcal{V}'_1 = \mathcal{K}_2 \parallel \mathcal{V}_2 \parallel 0^{t'}$. As we have discussed in Case I.4.d, if $\mathcal{K}_2$ is longer, no restriction is required for $\mathcal{K}_2$. However, if $\mathcal{K}_1$ is longer, we must have the last $\max(0, t' - \mathcal{L}_{V'_1})$ of $\mathcal{K}_1$ to be all zero. Now in here we have $\mathcal{L}_{V'_1} = \mathcal{L}_{V_1} - (8 - t)$. So this is the only difference of the restriction we have for this family from the restriction in case I.4.d.