

IMPROVED DIFFERENTIAL ATTACKS ON RECTANGLE

A THESIS SUBMITTED TO  
THE GRADUATE SCHOOL OF INFORMATICS  
OF  
MIDDLE EAST TECHNICAL UNIVERSITY



BY

ASUMAN ŞENOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR  
THE DEGREE OF MASTER OF SCIENCE  
IN  
CYBER SECURITY

AUGUST 2017



Approval of the thesis:

**IMPROVED DIFFERENTIAL ATTACKS ON RECTANGLE**

submitted by **ASUMAN ŞENOL** in partial fulfillment of the requirements for the degree of **Master of Science in Cyber Security Department, Middle East Technical University** by,

Prof. Dr. Deniz Zeyrek Bozşahin  
Dean, Graduate School of **Informatics**

\_\_\_\_\_

Assist. Prof. Dr. Aybar Can Acar  
Head of Department, **Cyber Security**

\_\_\_\_\_

Assist. Prof. Dr. Aysu Betin Can  
Supervisor, **Informatics Institute, METU**

\_\_\_\_\_

Dr. Cihangir Tezcan  
Co-supervisor, **Department of Mathematics, METU**

\_\_\_\_\_

**Examining Committee Members:**

Assoc. Prof. Dr. Ali Doğanaksoy  
Department of Mathematics, METU

\_\_\_\_\_

Assist. Prof. Dr. Aysu Betin Can  
Informatics Institute, METU

\_\_\_\_\_

Assist. Prof. Dr. Aybar Can Acar  
Informatics Institute, METU

\_\_\_\_\_

Assist. Prof. Dr. Cengiz Acartürk  
Informatics Institute, METU

\_\_\_\_\_

Prof. Dr. Ali Aydın Selçuk  
Computer Engineering Department, TOBB ETU

\_\_\_\_\_

**Date:**

\_\_\_\_\_





**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

Name, Last Name: ASUMAN ŞENOL

Signature :

# ABSTRACT

## IMPROVED DIFFERENTIAL ATTACKS ON RECTANGLE

Şenol, Asuman

M.S., Department of Cyber Security

Supervisor : Assist. Prof. Dr. Aysu Betin Can

Co-Supervisor : Dr. Cihangir Tezcan

August 2017, 93 pages

Differential attacks aim to capture the round keys by examining the changes in the output when a small change is applied to the input. This method is based on examining the differential behavior of the cryptosystem and guessing the affected round keys by using candidate plaintext and ciphertext pairs. It was shown that it may not be possible for the attacker to fully uncover the guessed keys. This situation occurs when the cipher contains S-boxes after the key addition layer and the guessed keys have a specific difference for a fixed S-box output difference for some S-boxes. Such an S-box property is called a differential factor. Because of the uncovered keys which is caused by not taking differential factors into account, attacks in the literature obtained by theoretical methods may not work correctly in practice. In addition to that, more powerful differential attacks can be proposed with undisturbed bits because these bits provide discovering longer differential characteristics. As these attacks are corrected by considering differential factors and undisturbed bits, the claimed time complexity may increase or decrease. RECTANGLE and PRESENT are two lightweight block ciphers with SPN structure and their S-boxes have differential factors and undisturbed bits. In this work, we corrected previously published differential attacks on RECTANGLE and on PRESENT by the help of undisturbed bits and we showed that these attacks can actually be performed with time complexities reduced with the help of differential factors and undisturbed bits .

Keywords: differential cryptanalysis, block cipher, lightweight, differential factor



# ÖZ

## RECTANGLE ALGORİTMASINDA GELİŞTİRİLMİŞ DİFERANSİYEL SALDIRILAR

Şenol, Asuman

Yüksek Lisans, Siber Güvenlik Bölümü

Tez Yöneticisi : Yrd. Doç. Dr. Aysu Betin Can

Ortak Tez Yöneticisi : Dr. Cihangir Tezcan

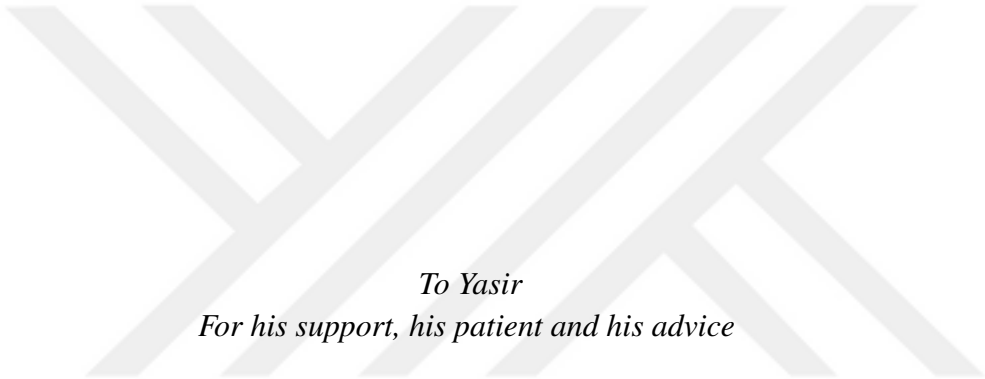
Ağustos 2017 , 93 sayfa

Diferansiyel saldırılar, girdiye küçük bir değişiklik uygulandığında çıktıdaki değişiklikleri inceleyerek raund anahtarlarını yakalamayı amaçlar. Bu yöntem, kriptosistemin diferansiyel davranışını incelemek ve etkilenen raund anahtarlarını, aday düz metin ve şifreli metin çiftlerini kullanarak tahmin etmek üzerine kuruludur. Saldırganın tahmin edilen anahtarları tam olarak elde etmesi bazı durumlarda mümkün değildir. Bu durum, şifre ekleme katmanının ardından S kutuları içerdiğinde ve tahmin edilen anahtarların, bazı S kutuları için sabit bir S kutusu çıktı farkı için belirli bir farkı olduğunda ortaya çıkar. Böyle bir S-box özelliği, diferansiyel faktör olarak adlandırılır. Farklı diferansiyel faktörlerin dikkate alınmamasıyla ortaya çıkan açık anahtarlar ile teorik yöntemlerle elde edilen saldırılar uygulamada doğru şekilde çalışmayabilir. Bu özelliğe ek olarak, daha güçlü ataklar karıştırılmamış bitler ile yapılabilir çünkü bu bitler daha uzun diferansiyel karakteristiklerin keşfedilmesini sağlarlar. Bu saldırılar, farklı faktörleri ve karıştırılmamış bitleri göz önüne alarak düzeltildiğinde, iddia edilen zaman karmaşıklığı artabilir veya azalabilir. RECTANGLE ve PRESENT SPN yapısında hafif blok şifrelerdir ve S kutuları diferansiyel faktörlere ve karıştırılmamış bitlere sahiptir. Bu çalışmada, önceden RECTANGLE ve PRESENT algoritmalarına yapılan, yayınlanmış diferansiyel saldırıları karıştırılmamış bitler yardımı ile düzelttik ve bu saldırıların differansiel bitler yardımı ile zaman karmaşıklıklarının azaltılmasıyla gerçekleştirilebileceğini gösterdik.



Anahtar Kelimeler: diferansiyel kriptanaliz, blok Őifre, hafif Őifre, diferansiyel faktör





*To Yasir  
For his support, his patient and his advice*

## ACKNOWLEDGMENTS

I would like to thank my supervisor Assist. Prof. Dr. Aysu Betin Can and co-supervisor Dr. Cihangir Tezcan for their constant support, guidance and friendship. While I was writing the thesis, Dr. Tezcan spared all his support even though he was not in Turkey. Dr. Tezcan always supported me like he was in Turkey. It was a great honor to work with him for the last two years and our cooperation influenced my academical life highly.

I have learned many theories in the courses of Master Degree Program, I have also had the opportunity to try these theories practically while working. During this time, all my colleagues supported and encouraged me to accomplish my master's degree successfully. Also, I would like to thank my team leaders for helping me to go to school at any time.

Lastly, sincerest thanks to my mom Gülay Şenol, my dad Yıldırım Şenol, my brother Can Şenol. Throughout my entire life, they have supported me materially and morally. I especially thank to my fiance Yasir Yazıcı, he is with me with his love, his knowledge and his experience at every moment. He supports and believes in me all the way through my academic life.

## TABLE OF CONTENTS

ABSTRACT . . . . .	vi
ÖZ . . . . .	viii
ACKNOWLEDGMENTS . . . . .	xi
TABLE OF CONTENTS . . . . .	xii
LIST OF TABLES . . . . .	xvii
LIST OF FIGURES . . . . .	xix
LIST OF ABBREVIATIONS . . . . .	xx

### CHAPTERS

1	INTRODUCTION . . . . .	1
1.1	Contributions of the Thesis . . . . .	2
1.2	Outline . . . . .	3
2	BACKGROUND ON CRYPTOGRAPHY . . . . .	5
2.1	Foundation of Cryptography . . . . .	5
2.2	Evolution of Cryptography . . . . .	5
2.3	Characteristics of Modern Cryptography . . . . .	6
2.4	Context of Cryptography . . . . .	6

2.5	Purposes of Cryptography . . . . .	7
2.6	Types of Cryptography . . . . .	8
2.6.1	Secret Key Cryptography . . . . .	8
2.6.2	Public Key Cryptography . . . . .	8
2.6.3	Hash Functions . . . . .	9
3	BLOCK CIPHERS . . . . .	11
3.1	Types of Block Ciphers According to Their Structure . . . . .	11
3.2	Modern Block Cipher Types . . . . .	12
3.3	Cryptanalysis of Block Ciphers . . . . .	14
3.4	Lightweight Block Ciphers . . . . .	15
3.4.1	Design of Lightweight Block Ciphers . . . . .	15
3.4.2	The Most Widely Known Lightweight Block Ciphers . . . . .	17
3.4.2.1	PRESENT . . . . .	17
3.4.2.2	PRINTCIPHER . . . . .	18
3.4.2.3	DESL, DESX and DESXL . . . . .	19
3.4.2.4	LED . . . . .	20
3.4.2.5	KATAN and KTANTAN . . . . .	20
3.4.2.6	KLEIN . . . . .	21
3.4.2.7	LBLOCK . . . . .	22
3.4.2.8	TWINE . . . . .	23
3.4.2.9	EPCBC . . . . .	23

	3.4.2.10	PUFFIN . . . . .	24
4		S-BOXES . . . . .	27
	4.1	Introduction of S-Box . . . . .	27
	4.2	S-Box Properties . . . . .	27
		4.2.1 Differential Uniformity . . . . .	27
		4.2.2 Robustness . . . . .	28
		4.2.3 Non-linearity . . . . .	28
		4.2.4 Balancing . . . . .	28
		4.2.5 Strict Avalanche Criterion (SAC) . . . . .	29
		4.2.6 Branch Number . . . . .	29
		4.2.7 Undisturbed Bits . . . . .	29
		4.2.8 Differential Factors . . . . .	30
	4.3	Differential Factors and Cryptanalysis . . . . .	31
5		ATTACKS ON BLOCK CIPHERS . . . . .	33
	5.1	Cryptanalytic Attacks . . . . .	35
		5.1.1 Differential Cryptanalysis . . . . .	35
		5.1.1.1 Overview of Basic Attack . . . . .	37
		5.1.1.2 Extracting Key Bits . . . . .	37
		5.1.1.3 Complexity of the Attack . . . . .	38
		5.1.2 An Example of Differential Cryptanalysis: Differ- ential Cryptanalysis of UltraLightweight PRESENT Cipher . . . . .	38

5.2	Corrected Attack on PRESENT . . . . .	40
6	OVERVIEW OF RECTANGLE . . . . .	43
6.1	The Last Version of The RECTANGLE Algorithm . . . . .	44
6.1.1	Representation of Plaintext and Subkey as Matrix .	44
6.1.2	Substitution and Permutation Operations . . . . .	45
6.1.2.1	S-box . . . . .	45
6.1.2.2	Differential Factors . . . . .	46
6.1.3	Key Schedule . . . . .	46
6.1.4	Whole Cipher . . . . .	48
6.2	The REC-0(First Version of the RECTANGLE Algorithm) . . . . .	49
6.2.1	Representation of Plaintext and Subkey as Matrix .	49
6.2.2	Substitution and Permutation Operations . . . . .	50
6.2.2.1	S-box . . . . .	50
6.2.2.2	Differential Factors . . . . .	50
6.2.3	Key Schedule . . . . .	51
6.2.4	Whole Cipher . . . . .	51
6.3	Differential Attacks on RECTANGLE . . . . .	51
6.3.1	19-round Related-key Differential Attack on REC-0	52
6.3.1.1	Differential Characteristics . . . . .	52
6.3.1.2	A related-key differential attack on REC-0 with key length 80 . . . . .	54

6.3.2	18-round Related-key Differential Attack on RECT- ANGLE . . . . .	61
7	CORRECTIONS AND IMPROVEMENTS ON DIFFERENTIAL AT- TACKS ON RECTANGLE . . . . .	63
7.1	Improvement on 19-Round Related-Key Differential Attack on REC-0 . . . . .	63
7.2	Experimental Differential Cryptanalysis . . . . .	64
7.3	Improvement on 18-Round Differential Attack on RECTANGLE	69
8	CONCLUSION . . . . .	71
	REFERENCES . . . . .	73
	APPENDICES	
A	APPENDIX I . . . . .	81



## LIST OF TABLES

### TABLES

Table 3.1	Sample Substitution Table . . . . .	13
Table 3.2	Sample Permutation Table . . . . .	14
Table 3.3	S-box of PRESENT Cipher . . . . .	17
Table 3.4	S-box of PRINTCIPHER . . . . .	19
Table 3.5	S-box of DESL and DESXL Cipher . . . . .	19
Table 3.6	S-box of LED Cipher . . . . .	20
Table 3.7	S-box of KLEIN Cipher . . . . .	21
Table 3.8	S-box of LBLOCK Cipher . . . . .	23
Table 3.9	S-box of EPCBC Cipher . . . . .	24
Table 3.10	S-box of PUFFIN Cipher . . . . .	25
Table 3.11	Comparison of lightweight block ciphers . . . . .	26
Table 4.1	Undisturbed bits of RECTANGLE Cipher . . . . .	30
Table 5.1	Differential distributions of the S-box in PRESENT . . . . .	39
Table 5.2	Differential Factors of PRESENT . . . . .	39
Table 5.3	16-round differential-linear attack of [62]. Values that need to be obtained are shown in bold. . . . .	40
Table 6.1	S-box of RECTANGLE . . . . .	45
Table 6.2	Differential Factors of RECTANGLE . . . . .	46
Table 6.3	Round Constants of Rectangle Cipher . . . . .	47

Table 6.4	S-box of REC-0 . . . . .	50
Table 6.5	Differential Factors of the S-box of REC-0 . . . . .	50
Table 6.6	Differential distributions of the S-box in REC-0 . . . . .	53
Table 6.7	n-round differential characteristics with fewer active S-boxes and the differences of the 2nd round and the 16th round subkeys are . . . . .	53
Table 6.8	All 15-round differential characteristics with 26 – 30 active S-boxes	54
Table 7.1	19-round differential-linear attack of REC-0. differential factors are shown in bold. . . . .	63
Table 7.2	19-round differential-linear attack of REC-0. Bits that should be captured because of undisturbed bit in the 1 <sup>st</sup> are shown in bold. . . . .	69
Table 7.3	The input difference and the output difference of the 14-round dif- ference propagation . . . . .	69

## LIST OF FIGURES

### FIGURES

Figure 3.1	The Structure of Feistel Ciphers . . . . .	12
Figure 3.2	The Structure of SPN Type of Ciphers . . . . .	13
Figure 3.3	The Structure of PRESENT Cipher . . . . .	18



## LIST OF ABBREVIATIONS

DDT	Difference Distribution Table
IC	Integrated Circuit
IoT	Internet of Things
LFSR	Linear Feedback Shift Register
MILP	Mixed-Integer Linear Programming
RC	Round Constant
REC-0	The old version of RECTANGLE Cipher
SPN	Substitution Permutation Network

# CHAPTER 1

## INTRODUCTION

People have kept secrets for centuries for different reasons like private communication, secure commerce and so on. For this purpose, they created different methods like changing the original content of a message with numbers, symbols and pictures in history. This way, they were able to assure that the content of their message can only be seen by the authorized people. This was the initial state of encryption. Since then, the encryption has evolved and become more complex over the centuries. Now, encryption can be done by advanced mathematical techniques with the beginning of the Information Age.

There are continuous changes in today's information technology environment. At the beginning of Information Age, information technology systems were based on host technology that a supercomputer served many users through thin clients. After these times, one user - one computer model which means each user has his own computer became popular. This model is still widely used today. On the other hand, we have observed that there is a growing interest in controlling several electronic devices that interact with each other through the network by one user. This is one user-many computer model which is commonly known as "Internet of Things (IoT)" or "Smart Object" networks. RFID tags, sensors, contact-less smart cards, health-care devices are some examples of IoT devices and as the number of IoT devices increase, security emerges as a growing problem every day. Cyber Security has gained importance with the improvement of the technology especially with the development of IoT technologies and it aims to protect devices from unauthorized access.

The security in IoT devices is provided by cryptography as are in the other devices.

However, IoT devices generally operate on platforms with limited resources and limited computing power. Because of those reasons, modern cryptographic algorithms that is suitable for personal computers are not suitable for IoT devices. Thus, the need for the algorithms that is used in IoT devices which require less power, less energy, less cost, and less memory has arisen. To meet this need, lightweight cryptographic algorithms are used.

Mostly block ciphers are used in lightweight cryptography and these block ciphers have exposed to several attacks. Differential cryptanalysis is one of the most important attack type to capture key bits. Since differential cryptanalysis are usually theoretical, various calculation errors arise when they are not tested in practice. In this thesis, we study the problems that arise when the theoretical differential attacks are not practically tested.

## **1.1 Contributions of the Thesis**

One user - many computer model developed quickly in the last ten years and this development brought new challenging problems at the same time. The main reason for these problems are related to the physical and economic constraints of them. In other words, conventional encryption techniques are not suitable for these devices since they have limitations. This indecency has accelerated the development of the lightweight cryptography.

A lot of lightweight ciphers have emerged along with the rapid developments in this area as we have explained in Section 3.4.2. As different lightweight ciphers are proposed, many differential attacks against to these ciphers have been done and still continued to be done. However these attacks are mostly theoretical and they are not checked practically. Dr. Tezcan noticed the presence of differential factors and undisturbed bits [55], when he investigated these theoretical attacks. At the end of these researches, the following result has been achieved: if differential factors and undisturbed bits are not considered by the attackers, the attacks in the literature obtained by theoretical methods may not work correctly in practice. When these attacks are corrected by considering them, the claimed time complexity may increase or decrease

due to differential factors.

In this thesis, we have investigated RECTANGLE and PRESENT cipher to see if it contains a differential factor and a undisturbed bit. We started our research by examining the whole ciphers to understand the 18-round [69], and 19-round [45] differential attacks on RECTANGLE [69] block cipher and 16-round [62] differential attack on PRESENT. Consequently, the time complexity of the 19-round related-key differential attack of [45] increased by a factor of  $2^{1.07}$  with the help of two differential factors and we showed that 3 more bits of the key should be guessed because of the undisturbed bits. In addition to that, we showed that the 16-round differential attack of [62] on PRESENT needs to guess 8 more bits of the key due to the undisturbed bits. These observations and improvements were also published in [57] and [56].

## 1.2 Outline

In the first chapter, we introduce you about our motivation. We also define the problem that we want to solve in this thesis and explain the contribution of the thesis. In the second and third chapters, we give a short introduction about cryptographic primitives and block ciphers, respectively. Chapter 4 gives an overview of S-boxes and the properties of it. In Chapter 5, we discuss the attack types of block ciphers. Moreover, after investigating the structure of RECTANGLE [69] in Chapter 6, we investigate the 18-round [69], and 20-round [45] differential attacks on RECTANGLE [69] block cipher and we provide some corrections for these attacks considering differential factors in Chapter 7. Chapter 8 summarizes the dissertation.





## CHAPTER 2

### BACKGROUND ON CRYPTOGRAPHY

In this chapter, we give brief information about cryptography.

#### 2.1 Foundation of Cryptography

The foundations of cryptography is based on the Roman and Egyptian civilizations. The first known example of cryptography is 'hieroglyph'. About 4000 years ago, the Egyptians communicated with the help of hieroglyphic messages in a hidden way. It consists of more than 700 signs and each sign represents either a particular object or a specific voice as described in [36].

#### 2.2 Evolution of Cryptography

During and after the European Renaissance, different cryptographic techniques were used at various Italian and Papal states. In addition, several analysis and attack techniques had been explored to break the secret codes in this period. The milestones of this field are as follows:

- New coding techniques such as Vigenere Cipher were found in the 16th century. The Vigenere Cipher was developed by Blaise de Vigenere and it is a simple form of polyalphabetic substitution.
- Only after the 19th century, cryptography has become more sophisticated than transient approaches to cryptography. Morse code play fair, beale ciphers are

found and used in this century.

- In the early 20th century, mechanical and electromagnetical machines were discovered such as the Enigma rotor machine. They provided more efficient way for encoding.
- During World War II, both cryptography and cryptanalysis became based on advanced mathematics.

With this progress, government agencies, military units and some institutional houses began to adopt using cryptography to protect their secrets from others. Now, with the advent of computers and the Internet, modern cryptography methods are used by the people all over the world.

### **2.3 Characteristics of Modern Cryptography**

Modern cryptographic algorithms work on binary bits rather than letters and sequences as in traditional methods. In addition to that, the coding algorithms are not kept as a secret in modern cryptography. Confidentiality is achieved by only using a secret key. The difficulties on calculation make it difficult for an attacker to obtain the original information even if he knows the algorithm used in coding.

### **2.4 Context of Cryptography**

The study of cryptosystems can be divided into two categories.

- **Cryptography** : Cryptography is the art and science of concealing the messages to provide security. This word is the combination of two Greek words; 'Krypto' means hidden and 'graphene' means writing. Cryptography ensures the safety of data. It is generally based on mathematical algorithms. One can think of cryptography as the establishment of a large set of tools with different techniques in security applications.

- **Cryptanalysis** : Cryptanalysis is the science of analyzing the security in the ciphers by finding the weaknesses in design. Cryptanalysis is the sibling of cryptography and they are inseparable fields. When a new encryption technique is designed, its cryptanalysis results are published with it too.

## **2.5 Purposes of Cryptography**

The purposes that are intended to be fulfilled by cryptography is as follows:

### **1. Confidentiality**

Confidentiality is the most fundamental security purpose of cryptography. It aims to keep information away from an unauthorized person. Confidentiality can be achieved in numerous ways, from providing physical security to using mathematical algorithms for data encryption.

### **2. Data Integrity**

Data Integrity aims to detect any changes in the data. The data may be altered intentionally or accidentally by an unauthorized person. Integrity confirms that the data has not been changed since it was last created, transmitted or stored by an authorized user. Data integrity can not prevent data from being altered, but it can detect whether data is being manipulated by an unauthorized user or not.

### **3. Authentication**

Authentication provides identification of the sender. Receiver can be sure that the received data is only sent by the verified sender with the help of the authentication.

### **4. Non-Repudiation**

Non-Repudiation provides that an entity cannot deny the his previous commitments or actions. Non repudiation is the most desirable feature when there is a disagreement about a data exchange especially while making an online payment.

## 2.6 Types of Cryptography

There are three cryptography techniques which are secret key cryptography, public key cryptography and hash functions.

### 2.6.1 Secret Key Cryptography

This encryption technique uses only one key. Sender and recipient applies a key to encrypt and decrypt messages with the same key. Since only single key is used for both encryption and decryption in this technique, it can also be called symmetric encryption. Key distribution is the biggest problem in this technique and this problem can be solved by asymmetric cryptography.

There are two main types of symmetric algorithms which are Block Ciphers and Stream Ciphers.

A stream cipher is an encryption algorithm that encrypts each binary digit in a data stream with a key. *A5/1* [21] is the most widely used stream cipher algorithm in GSM in order to provide air communication privacy. *RC4* [52] and *E0* [19] are the other examples of stream ciphers and are used in web and blue-tooth communication, respectively.

A block cipher is an encryption algorithm that encrypts a fixed size of  $n$ -bits of data at one time. 64 bits, 128 bits, and 256 bits of blocks can be used. Block ciphers are the main scope of this thesis and we inform you about the block ciphers in Chapter 3 in detail.

### 2.6.2 Public Key Cryptography

It is suggested by Diffie & Hellman in 1976 [18]. It ensures that the people who have never been seen and talked before can communicate securely in an unsecured channel. There are two keys in this type of cryptography; one public, one private. While public key is known everyone, private key is kept secret; i.e., only owner knows it. All messages (text, binary files, or documents) are encrypted using the public key.

These can only be decrypted using the matching private key, but only by applying the same algorithm.

It provides solutions of authentication, non repudiation problems unlike in secret key cryptography. However, it is slower than symmetric encryption. In addition to that, much more processing power is required to encrypt and decrypt the content of the message.

RSA [43], Diffie-Hellman [18], Digital Signature Algorithm, ElGamal [20] are the examples of asymmetric cryptography usage.

### 2.6.3 Hash Functions

A hash function maps variable length inputs to fixed length output(buckets). Block length can be 128 or 192 bits. Hash functions are used in many parts of cryptography like digital signatures, message authentication codes, pseudo random number generators, key derivation functions. There are many different types of hash functions with different security features. These features are as follows:

1. Collision Resistance : The most basic security property of a hash function is collision-resistance, which suggests that it should be hard to find two inputs  $x_1$  and  $x_2$  such that  $h(x_1) = h(x_2)$  where  $h$  is the hash function. If a person can find collision easily, he can get a authentic digitally signed message, find a different message that produces the same digest (collision), and then he can use the same signature as the fake message. Thus, someone trying to verify the signature can not recognize the difference. The cost of finding a collision is  $2^{n/2}$  where  $n$  is the number of bits.
2. Preimage Resistance : This property suggests that it should be hard to find any input  $x$  such that  $h(x) = y$  for a given output  $y$ . A generic attack would require around  $2^n$  operations where  $n$  is the number of bits.
3. Second Preimage Resistance : This property suggests that it should be hard to find any other input  $x_2$  such that  $h(x_2) = y$  for given an output  $y$  and input  $x_1$  such that  $h(x_1) = y$ . A second-preimage is also a collision but finding a second-

preimage requires around more operations than collision, i.e.,  $2^n$  operations where  $n$  is the number of bits.



## CHAPTER 3

### BLOCK CIPHERS

In a block cipher, plaintext information is divided into fixed size of  $n$ -bits of data which are called blocks. Moreover, encryption for block ciphers consists of several rounds which contains various operations. Each block is encrypted at a time, by repeating the round function in each round. When each block is encrypted, they are combined together to obtain ciphertext.

While encrypting, all block sizes have to be same because otherwise cipher text message cannot be uniquely decrypted to a single plaintext block with one to one mappings. If a block is less than the block size, several methods can be used to equalize the block length like ciphertext stealing, padding.

#### 3.1 Types of Block Ciphers According to Their Structure

There are two main variations of the block cipher both of which aim to increase the security of encrypted text.

- **Product Ciphers** Product ciphers consists of multiple operations such as shifting, substituting, permutation. The combination could provide more powerful cryptosystem than other one that is used alone.
- **Iterated Block Ciphers** An iterated block cipher involves different numbers of repeated round functions. The idea behind of it is making cipher more secure by repeatedly using round functions.

### 3.2 Modern Block Cipher Types

Modern block ciphers consist of multiple operations iteratively so they have both product and iterated block cipher features.

- **Feistel Ciphers :** Feistel ciphers are a special class of iterated - product block ciphers. Plaintext is encrypted by splitting them into two halves. The round function  $f$  is applied to one half using a subkey and the output of  $f$  is exclusive-ored with the other half as shown in Figure 3.1. The two halves are then swapped. Each round follows the same pattern except for the last round where there is no swap.

DES, TEA [65], XTEA [38], RC5 [42], RC6 [14], CAMELLIA [2] are the examples of Feistel ciphers.

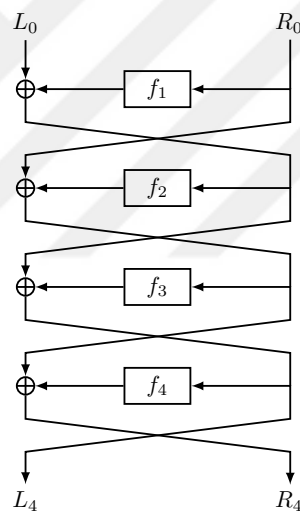


Figure 3.1: The Structure of Feistel Ciphers

- **Substitution-Permutation Network Ciphers :** SPN is a highly organized way of constructing a product-iterated cipher. Each round consists of three operations; substitution, transposition of the bits (i.e., permutation of the bit positions), and key mixing. As presented in Figure 3.2,

1. Firstly, plaintext blocks are XORed with the key,
2. Then substitution is applied to obtained bits,
3. After that operation, the positions of bits are permuted,



4. Lastly, one more key XOR can be applied to obtained bits.

There are a few SPN ciphers; AES [17], the AES finalist SERPENT [5], the light weight block cipher PRESENT [9], RECTANGLE [68] and etc. The operations to provide better security are as follows:

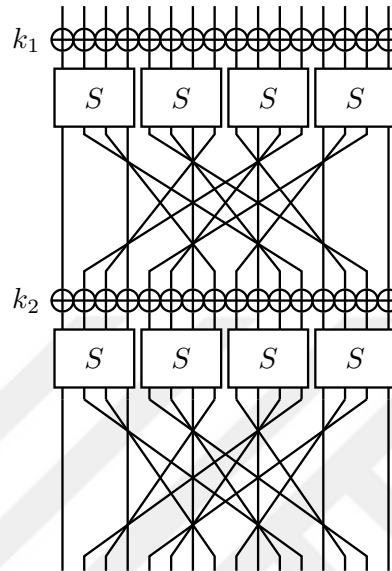


Figure 3.2: The Structure of SPN Type of Ciphers

1. Substitution : Substitution operation is generally provided by S-boxes. In general, an S-box takes some number of input bits and transforms them into some number of output bits. When one input bit changes, about half of the output bits should also change in a good S-box. In Chapter 2, S-box is introduced.

Table 3.1: Sample Substitution Table

input	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
output	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

2. Permutation : The permutation operator, P-box, is a permutation of all bits, it transposes the bits or permutes the bit positions, and then feeds them to the next round S boxes. A good P-box has the property that the output bits of any S-box are distributed to as many S-box inputs as possible.

For security, it is expected that for any chosen key, a good block cipher acts like a randomly selected permutation. Besides, it is expected no relation between permutations that are obtained by keys that are related somehow.

Table 3.2: Sample Permutation Table

input	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
output	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

3. Key-mixing : In key-mixing operation, generally bitwise exclusive-OR is used between the key bits(associated with round-subkey) and the data block input. The subkey is produced by the cipher's master key with a process that is known as **key schedule**.

### 3.3 Cryptanalysis of Block Ciphers

Security in block ciphers is based on only the secrecy of the key. The person who has captured the key will also get secure information. There are various methods used to capture this secret key. The most obvious method is to try to decrypt ciphertext with all the possible keys, which is known as exhaustive search or brute force. This attack is done by obtaining a plaintext ciphertext pair and encrypting plaintext with every possible keys. While encrypting, if the encrypted text matches with the previously obtained ciphertext, then that key is determined as the correct key. The key space can be enlarged to protect the cryptosystem against this attack. For n-bit key,  $2^n$  operations are required to perform an exhaustive search.

Table attack is the another method to capture key which requires less operations than exhaustive search. In this method, a database is used to store all plaintext, ciphertext pairs. A database query is used to find matching plaintext that corresponds ciphertext. However, if the size of the file which contain plaintext,ciphertext pairs can be very large, query execute times will be very high.

To say that a cipher is broken, it has to proposed an attack that requires less opera-

tions exhaustive search and less data than table attack. One of attack type to reduces data and time requirements is Time-Memory Trade-Off attack, which is developed by Hellman in [24]. The aim is performing exhaustive search and storing only a small portion of the resulting tables in a short way.

Differential cryptanalysis is the other technique to break block ciphers by investigating the relation between the input and the corresponding output differences. However, it may not be possible fully capture the attacked round key bits if the active S-box of cipher has differential factors as described in [58]. In our study, we have investigated the effects of differential factors on RECTANGLE cipher. Differential cryptanalysis and differential factors are explained in detail in the following chapters.

### **3.4 Lightweight Block Ciphers**

Embedded systems are deployed in various domains, including industrial installations, critical and nomadic environments, private spaces and public infrastructures. Their operation typically involves access, storage and communication of sensitive and/or critical information that requires protection, making the security of their resources and services an imperative design concern. The demand for applicable cryptographic components is therefore strong and growing. However, the limited resources of these devices, in conjunction with the ever-present need for smaller size and lower production costs, hinder the deployment of secure algorithms typically found in other environments and necessitate the adoption of lightweight alternatives.

#### **3.4.1 Design of Lightweight Block Ciphers**

While traditional cryptography focuses on the solutions by ignoring the requirements of constrained devices, lightweight cryptography mainly focuses on designing scheme for devices constrained capabilities in power supply, connectivity, hardware and software. There are some design criteria that must be carefully investigated to meet the required design goals. Some of the design criteria examples are memory consumption, power consumption, latency, chip area, cost of one implementation, side channel resistance and throughput.

While designing lightweight block ciphers, there are four steps which are: specification, design, implementation and cryptanalysis.

1. In specification step, main design criteria is defined with required threshold value. Specifying design criteria depends mostly on platform where algorithm is implemented.
2. The second step is designing. Block cipher is designed with respect to the design criteria that is defined in previous step. New cipher is designed by optimizing traditional cryptographic algorithms such as DES, RSA [43] or designed from scratch. In the former case, confidential algorithms should be selected carefully while optimizing the certain parameters since this change may enhance complexity or compromise security. During design of lightweight block ciphers,
  - (a) Smaller block size can be used. In this way, memory can be saved. Also using small block size reduces the limits on the maximum number of plaintext blocks to be encrypted. For instance,  $2^{32}$  blocks are used to distinguish outputs of a 64-bit block cipher from a random sequence.
  - (b) Smaller key size can be used for efficiency.
  - (c) Rounds can be made simple. To achieve this, 4-bit S-boxes, bit permutations (like in PRESENT [9]) and recursive MDS matrices (like in LED [23]) are preferred. For better security, round numbers can be increased.
  - (d) Simple key schedules can be used. Complex key schedules increases the memory, latency and the power consumption of implementation. To avoid them, simple key schedules that can generate sub-keys on the fly can be used. On the other hand, using simple key schedule results as related key, weak keys, known keys or chosen keys attacks. To avoid these attacks, a secure derivation function can be used like in [61].
  - (e) Minimal implementations can be used. Some applications may require encryption operations, others need only decryption. Implementing only necessary functions of a cipher requires less resources than implementing the full cipher.

3. In the implementation step, lightweight cipher is implemented as specified in the design step. Implementation cost should be considered in this step and if the cost is very high then some changes in the specification and design steps can be made by returning these steps.
4. In the last step, lightweight cipher is tested by cryptanalysis to determine the security of cipher.

### 3.4.2 The Most Widely Known Lightweight Block Ciphers

#### 3.4.2.1 PRESENT

PRESENT [9] is the mostly known ultra-lightweight SPN type block cipher presented at CHES 2007 firstly. It has been designed for area and power constraints devices. It consists of 31 rounds. The block length is 64 bits and two key lengths of 80 and 128 bits are supported. S-box is 4 bits and used 16 times in one round.

Each round consists of the following 3 steps:

1. Subkey Addition: Key XORed with cipher.
2. Substitution Operation: Used 4 bits S-box.
3. Permutation Operation: Uses bit-wise permutation.

S-box used in PRESENT cipher is provided in Table 3.3:

Table 3.3: S-box of PRESENT Cipher

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S(x)	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Security of PRESENT Cipher :

The best known attack on PRESENT is provided in [62] We improved this attack in [57]. This improvement is introduced in Chapter 5 in detail.

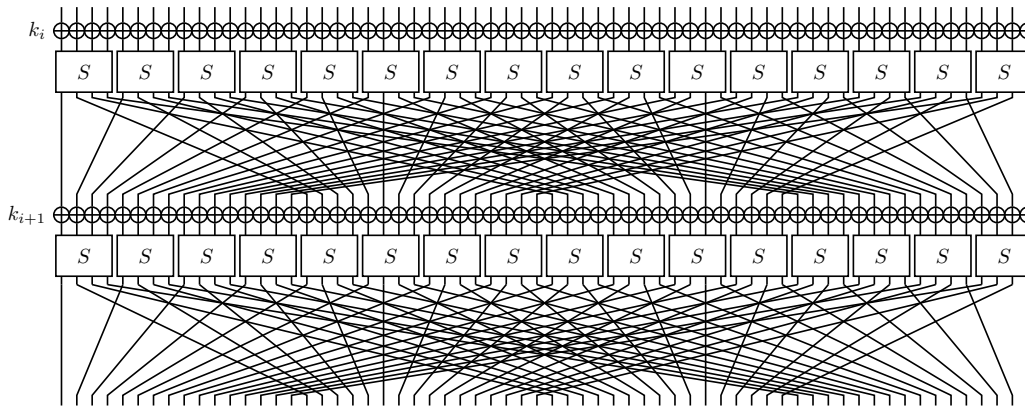


Figure 3.3: The Structure of PRESENT Cipher

### 3.4.2.2 PRINTCIPHER

PRINTCIPHER [31] is designed for integrated circuit (IC) printing. IC printing enables circuits to be produced and personalized at very low costs.

PRINTCIPHER is a SPN type of cipher. It consists of 48 or 96 rounds and each round also contains add key, substitution and permutation layers. It operates on 48-bit, 96-bit blocks with key-sizes of 80, 160 bits. It has 16 identical  $4 \times 4$  S-boxes.

PRINTCIPHER is a fixed key algorithm, i.e., key schedule is not used; so there is no need to update key in each round. First the key is divided into two parts and the first part is XORed with the ciphertext and the second part is used for permutation before substitution.

Each round in PRINTCIPHER consists of the following 5 steps:

1. XORing key with cipher.
2. Applying linear diffusion.
3. XORing rightmost 6 bits with round-constant.
4. Permuting bits.
5. Mixing bits by using S-box.

S-box used in PRINTCIPHER is as shown in Table 3.4:

Table 3.4: S-box of PRINTCIPHER

x	0	1	2	3	4	5	6	7
S(x)	0	1	3	6	7	4	5	2

Security of PRINTCIPHER :

There are two attacks to break PRINTCIPHER in literature. The first one breaks less than half of the rounds as shown in [1]. The second is subspace attack which was presented in [34] and this breaks the full cipher. This attack can be seen as a weak-key variant of a statistical saturation attack. For such weak keys, a chosen plaintext distinguishing attack can be mounted in unit time.

**3.4.2.3 DESL, DESX and DESXL**

DESL [35] is the lightweight version of DES cipher. It consists of 16 rounds. It has a 56-bit key with a 64-bit block length based on Feistel network. Unlike the S-boxes usage on DES, DESL uses a single  $6 \times 4$  bits S-box 8 times. In addition, initial and final permutations are not used to reduce the cost and the area of block cipher in DESL.

DESX is secure a variant of DES cipher. It consists of 16 rounds. The block length is 64 bits and it accepts 184 bits key. S-box used in DESL and DESXL cipher is as presented in Table 3.5:

Table 3.5: S-box of DESL and DESXL Cipher

$S_0$	14	5	7	2	11	8	1	15	0	10	9	4	6	13	12	3
$S_1$	5	0	8	15	14	3	2	12	11	7	6	9	13	4	1	10
$S_2$	4	9	2	14	8	7	13	0	10	12	15	1	5	11	3	6
$S_3$	9	6	15	5	3	8	4	11	7	1	12	2	0	14	10	13

Security of DESL, DESX and DESXL Ciphers :

Up to now, there is no attack against DESL, DESX and DESXL in the literature.

#### 3.4.2.4 LED

LED is a SPN type of lightweight block cipher. The block length is 64 bits and two key lengths of 64 and 80 bits are supported. It consists of 32 rounds for a 64 bits key, 48 rounds for a 128 bits key. Each block of cipher is represented as a  $4 \times 4$  nibble matrix. It uses same inner transformation like on AES; but the transformation is optimized for hardware applications.

Key schedule is not used in LED. Instead of this, the key is XORed every 4 rounds. This deficiency is compensated by an increased number of rounds compared to AES. LED cipher uses S-box of PRESENT cipher as shown in Table 3.6.

Table 3.6: S-box of LED Cipher

x	0	1	2	3	4	5	6	7
S(x)	0	1	3	6	7	4	5	2

#### Security of LED Cipher :

There are two papers that investigate the security of LED. In [25], meet-in-the-middle technique is used to break 8 rounds of LED-64, and 16 rounds of LED-128 as shown. In [37], the authors show attacks for LED-64 reduced to 12 and 16 rounds.

#### 3.4.2.5 KATAN and KTANTAN

KATAN and KTANTAN are very efficient hardware oriented block ciphers, proposed at CHES 2009 [10]. They both accept 80-bits keys. Block sizes can be 32, 48 and 64 bits in this ciphers.

KTANTAN is more compact than KATAN and it is more suitable for fixed key devices.

The 80 bits key is loaded into a register which is repeatedly clocked in KATAN, whereas in KTANTAN, the key is fixed. Firstly, plaintext is divided into two parts. Then several bits are taken and executed in two non linear boolean functionn. Lastly the output of the boolean function is shifted to the left one bit. These operations are repated 254 times.



### Security of KATAN and KTANTAN Ciphers :

In [64], a meet-in-the-middle attack was proposed and it recovers the 80 bits secret key of the full round KTANTAN-32, KTANTAN-48, KTANTAN-64 with the time complexity  $2^{72.9}$ ,  $2^{73.8}$ ,  $2^{74.4}$ , respectively. Conditional differential is the best attack that has been mounted as described in [28], [29]. In [28], conditional differential cryptanalysis was proposed with practical complexity against KATAN-32 on 78 rounds, KATAN-48 on 70 rounds, KATAN-64 on 68 rounds. In [29], the authors of [28] extended the results in the related key settings against KATAN-32 on 120 rounds, KATAN-48 on 103 rounds, KATAN-90 on 68 rounds.

#### 3.4.2.6 KLEIN

KLEIN is a SPN type of lightweight block cipher. It works especially on legacy sensor platforms with very high performance and its hardware implementation is quite compact. Substitution layer composes of 4 bits S-box. The permutation layer consists of rotating and mixing operations. Block size of KLEIN is 64 bits and it accepts 64, 80, 96 bits key for 12, 16, 20 rounds, respectively. Each round in KLEIN Cipher consists of the following 4 steps:

1. AddRoundKey Operation: Bit wise XOR.
2. SubNibbles Operation: Nibbles are passed through the S-box.
3. RotateNibbles Operation: Left two bytes of nibbles rotated.
4. MixNibbles Operation: Same as MixColumns in AES.

S-box used in KLEIN cipher is as shown in Table 3.7:

Table 3.7: S-box of KLEIN Cipher

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S(x)	7	4	A	9	1	F	B	0	C	3	2	6	8	E	D	5

Key Schedule of KLEIN as follows:

1. Divide i-th subkey into two parts x,y where  $k_i = x.y$
2. Cycle left shift one bits in each part of key x, y ( $k_i = x \lll .y \lll$ )
3. Swap (x, y) such as  $(x, y) = (y, x \oplus y)$ .
4. XORed round-counter with  $3^{rd}$  byte of x and substitute  $2^{nd}$  and  $3^{rd}$  byte of y with S-box.

#### Security of KLEIN Cipher :

In [3], the authors achieved to break up to 14 and 15 rounds for KLEIN-80 and KLEIN-96, respectively.

#### **3.4.2.7 LBLOCK**

LBLOCK is a feistel type lightweight block cipher and it has been presented in [66]. The block length is 64 bits and two key lengths of 80 and 32 bits are supported for 32 rounds. Each round consists of 3 operations:

1. Subkey addition,
2. Substitution,
3. 4 bits permutation.

10 different  $4 \times 4$  bits S-boxes are used. 8 S-boxes for encryption, 2 S-boxes for key scheduling as shown in Table 3.8.

#### Security of LBLOCK Cipher :

Although LBLOCK has proposed recently, there are several attempts against it. In [63], biclique attack was presented against a full round version of LBLOCK with a complexity slightly lower than exhaustive key search. In [47], zero-correlation linear attack was introduced against LBLOCK by mounting a 22 rounds attack.

Table 3.8: S-box of LBLOCK Cipher

$S_0$	14	9	15	0	13	4	10	11	1	2	8	3	7	6	12	5
$S_1$	4	11	14	9	15	13	0	10	7	12	5	6	2	8	1	3
$S_2$	1	14	7	12	15	13	0	6	11	5	9	3	2	4	8	10
$S_3$	7	6	8	11	0	15	3	14	9	10	12	13	5	2	4	1
$S_4$	14	5	15	0	7	2	12	13	1	8	4	9	11	10	6	3
$S_5$	2	13	11	12	15	14	0	9	7	10	6	3	1	8	4	5
$S_6$	11	9	4	14	0	15	10	13	6	12	5	7	3	8	1	2
$S_7$	13	10	15	0	14	4	9	11	2	1	8	3	7	5	12	6
$S_8$	8	7	14	5	15	13	0	6	11	12	9	10	2	4	1	3
$S_9$	11	5	15	0	7	2	9	13	4	8	1	12	14	10	3	6

### 3.4.2.8 TWINE

TWINE is a Feistel type lightweight block cipher [51]. The block length is 64 bits and two key lengths of 80 and 128 bits are supported. It consists of 36 rounds whatever the key length is. The F- function is repeated 8 times per round; it is composed of a subkey addition and substitution operation.

#### Security of TWINE Cipher :

In [13], two biclique attacks are presented on TWINE-80 and on TWINE-128 with the time complexities  $2^{79.10}$  and  $2^{126.82}$ , respectively.

### 3.4.2.9 EPCBC

EPCBC is designed for electronic product code encryption. There are two types of EPCBC cipher which consists of 32 rounds. The first type of cipher has a 96-bit key with a 48-bit block length based on SPN and it is denoted EPCBC(48,96). On the other hand, the second type of cipher, denoted EPCBC(96, 96) has an 96-bit key with a 96-bit block length. Both of them use the structure of PRESENT cipher (PR-48 and PR-96) for encryption and they also use the S-box of PRESENT cipher. The only difference among them is key schedule. For EPCBC(48, 96), the left half of 96 bits key forms the first subkey and applies variant-Feistel structure for 8 rounds. In each variant Feistel structure is composed of 4 rounds of PR-96 cipher structure without key addition. For EPCBC(96, 96), all 96 bits of key is used as the first subkey and it applies 32 rounds of

PR-96 cipher structure without any key addition for 32 subkeys. S-box used in EPCBC cipher is as shown in Table 3.9:

Table 3.9: S-box of EPCBC Cipher

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S(x)	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

#### Security of EPCBC Cipher :

Using 32 rounds EPCBC ensures the security of the cipher against differential and linear cryptanalysis as explained in [67].

#### **3.4.2.10 PUFFIN**

PUFFIN [12] is an involutinal 32-rounds SPN type block cipher. Involutinal means that both encryption and decryption operations require same data path, in other words the same primitive is used for both encryption and decryption processes. It has an 128-bit key with a 64-bit block length. It has a simple key scheduling algorithm. Since it has low hardware complexity and good throughput, it is suitable for application specific integrated circuit (ASIC) and field programmable gate array (FPGA) technologies.

Each round of encryption / decryption consists of the following 3 steps:

1. Substitution S: Performs  $4 \times 4$  S-box 16 times in parallel.
2. Key addition K: Bit wise XOR.
3. Permutation P: Used P-layer.

S-box used in PUFFIN cipher is as presented in Table 3.10:

Key Scheduling for Encryption is as follows : Apply permutation P in all rounds and then take the inverse of the  $1^{st}$ ,  $2^{nd}$ ,  $3^{rd}$  and  $5^{th}$  bits in all rounds, except in  $2^{nd}$ ,  $5^{th}$ ,  $6^{th}$  and  $8^{th}$  rounds.

Table 3.10: S-box of PUFFIN Cipher

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S(x)	D	7	3	2	9	A	C	1	F	4	5	E	6	0	B	8

Key Scheduling for Decryption is as follows: Apply the inverse permutation P in all rounds and then take the inverse of the 30<sup>th</sup>, 62<sup>th</sup>, 71<sup>th</sup> and 120<sup>th</sup> bits in all rounds, except in 2<sup>nd</sup>, 5<sup>th</sup>, 6<sup>th</sup> and 8<sup>th</sup> rounds.

Security of PUFFIN Cipher :

In [33], it was showed that PUFFIN can be broken by differential cryptanalysis faster than exhaustive search by using less than the full code-book.

Comparison of lightweight block ciphers are as presented in Table 3.11:

Table 3.11: Comparison of lightweight block ciphers

Ciphers	Function	Architecture	Structure	Key	Block	Rounds
PRINTCIPHER-48 [31]	Enc.	Serialized	SPN	80	48	48
PRINTCIPHER-48 [31]	Enc.	Round-based	SPN	80	48	48
PRINTCIPHER-96 [31]	Enc.	Serialized	SPN	160	96	96
PRINTCIPHER-96 [31]	Enc.	Round-based	SPN	160	96	96
LED-64 [23]	Enc.	Serialized	SPN	64	64	32
LED-80 [23]	Enc.	Serialized	SPN	80	64	48
LED-96 [23]	Enc.	Serialized	SPN	96	64	48
LED-128 [23]	Enc.	Serialized	SPN	128	64	48
KTANTAN-32 [10]	Enc.	Serialized	LFSR	80	32	254
KTANTAN-48 [10]	Enc.	Serialized	LFSR	80	48	254
KTANTAN-64 [10]	Enc.	Serialized	LFSR	80	64	254
PRESENT-80 [9]	Enc.	Serialized	SPN	80	64	32
PRESENT-80 [9]	Enc.	Round-based	SPN	80	64	32
PRESENT-128 [9]	Enc.	Serialized	SPN	128	64	32
PRESENT-128 [9]	Enc.	Round-based	SPN	128	64	32
EPCBC-48 [67]	Enc.	Serialized	SPN	96	48	32
EPCBC-96 [67]	Enc.	Serialized	SPN	96	96	32
DES	Enc.	Serialized	Feistel	56	64	16
DESL [49]	Enc.	Serialized	Feistel	56	64	16
DESX [49]	Enc.	Serialized	Feistel	184	64	16
DESXL [49]	Enc.	Serialized	Feistel	184	64	16
TWINE-80 [13]	Enc.	Round-based	Feistel	80	64	36
TWINE-80 [13]	Enc.	Serialized	Feistel	80	64	36
TWINE-80 [13]	Enc+Dec	Round-based	Feistel	80	64	36
TWINE-128 [13]	Enc.	Round-based	Feistel	128	64	36
TWINE-128 [13]	Enc+Dec	Round-based	Feistel	128	64	36
PUFFIN [39]	Enc+Dec	Round-based	SPN	128	64	32
KLEIN-64 [22]	Enc+Dec	Round-based	SPN	64	64	12
KLEIN-80 [22]	Enc+Dec	Round-based	SPN	80	64	16
KLEIN-96 [22]	Enc+Dec	Round-based	SPN	96	64	20
KLEIN-64 [22]	Enc+Dec	Serialized	SPN	64	64	12
KLEIN-80 [22]	Enc+Dec	Serialized	SPN	80	64	16
KLEIN-96 [22]	Enc+Dec	Serialized	SPN	96	64	20
KATAN-32 [10]	Enc.	Serialized	LFSR	80	32	254
KATAN-48 [10]	Enc.	Serialized	LFSR	80	48	254
KATAN-64 [10]	Enc.	Serialized	LFSR	80	64	254
LED-64 [23]	Enc.	Serialized	SPN	64	64	32
LED-80 [23]	Enc.	Serialized	SPN	80	64	48
LED-96 [23]	Enc.	Serialized	SPN	96	64	48
LED-128 [23]	Enc.	Serialized	SPN	128	64	48
LBLOCK [66]	Enc.	Round-based	Feistel	80	64	32
LBLOCK [66]	Enc.	Serialized	Feistel	80	64	32
RECTANGLE-80 [69]	Enc.	Round-based	SPN	80	64	25
RECTANGLE-128 [69]	Enc.	Round-based	SPN	128	64	25

## CHAPTER 4

### S-BOXES

#### 4.1 Introduction of S-Box

The security of data is mostly dependent to substitution process in block ciphers. Substitution is a non linear transformation that provides confusion property as Shannon suggests in [46]. If substitution layer is not used, it is possible to recover the key using a simple Gaussian elimination, given a few known plaintext-ciphertext pairs. However, it is not sufficient for strong ciphers; diffusion property should also be included in them. Shannon also suggests in [46] that, strong ciphers could be built by combining substitutions with transposition repeatedly. In this chapter, we analyze S-boxes, the main element of the substitution process.

#### 4.2 S-Box Properties

For strong encryption, S-boxes have the following properties.

##### 4.2.1 Differential Uniformity

**Definition 4.2.1** [44] *Let  $F$  be an  $n \times s$  S-box where  $n \geq s$ . Let  $\delta$  be the largest value in differential distribution table of the S-box not counting the first entry in the first row namely*

$$\delta = \max_{a \in V_n, a \neq 0} \max_{\beta \in V_s} |\{x | F(x) \oplus F(x \oplus a) = \beta\}|$$

$\delta$  is called the differential uniformity of  $f$ ; i.e. the maximum value in a DDT.

S-box designers are inclined to keep differential uniformity low, since the differential cryptanalysis used with high differential probability.

#### 4.2.2 Robustness

**Definition 4.2.2** [44] Let  $F = (f_1, \dots, f_s)$  be an  $n \times s$  S-box, where  $f_i$  is a function on  $V_m$ ,  $i = \{1, \dots, s\}$  and  $n \geq s$ . Denote by  $L$  the largest value in the difference distribution table of  $F$ , and by  $N$  the number of nonzero entries in the first column of the table. In either case the value  $2^n$  in the first row is not counted. Then we say that  $F$  is  $R$ -robust against differential cryptanalysis, where  $R$  is defined by

$$R = \left(1 - \frac{N}{2^n}\right)\left(1 - \frac{L}{2^n}\right) \quad (4.1)$$

Differential uniformity is used as the first indicator for the strength of an S-box, whereas robustness gives more accurate information about the resistance to differential attack than differential uniformity. This is because discussion of robustness is not as easy as in differential uniformity.

#### 4.2.3 Non-linearity

**Definition 4.2.3**  $S : \{0, 1\}^x \rightarrow \{0, 1\}^y$  is defined as the least value of nonlinearity of all nonzero linear combinations of  $x$  boolean functions  $f_i : \{0, 1\} \rightarrow \{0, 1\}$ ,  $i = x - 1, \dots, 1, 0$ .

To increase S-box's power against linear cryptanalysis, nonlinearity of a S-box must be high.

#### 4.2.4 Balancing

An S-box with  $n$  input bits and  $m$  output bits, ( $m \leq n$ ), is balanced if each output occurs  $2^{n-m}$  times. For the S-box to be balanced it should have the same number of 0's and 1's.



#### 4.2.5 Strict Avalanche Criterion (SAC)

The strict avalanche criterion arises when an input bit is changed, and each output bit is changed by one half probability. The strict avalanche requires a significant change in the output vector if there is a slight change in the input vector. To achieve this effect, we need a function that has 50% dependency on each of the  $n$  input bits.

#### 4.2.6 Branch Number

**Definition 4.2.4** *The branch number of an  $n \times n$  S-box is*

$$BN = \min_{a \neq b} (wt(a \oplus b) + wt(S(a) \oplus S(b)))$$

where  $a, b \in F_n^2$  and the branch number can take the value of the smallest 2 for a bijective S-Box.

That property of a S-box is associated with algebraic attack and cube attacks. [15].

#### 4.2.7 Undisturbed Bits

**Definition 4.2.5** [54] *Depending on the design of an S-box, when a specific difference is given to the input (resp. output), difference of at least one of the output (resp. input) bits of the S-box may be guessed with probability 1. We call such bits undisturbed.*

Undisturbed bits of the output difference remain invariant for a specific input difference of a S-box, . For instance, when we examine the Difference Distribution Table of RECTANGLE cipher which is presented in Chapter 4, we can see that for a fixed input difference at last two bits do not change and these two bits are undisturbed bits as shown in Table 4.1.

Undisturbed bits can be used to find longer differential characteristics leading to more effective differential attacks. When we look at the literature, we can see that undis-

Table 4.1: Undisturbed bits of RECTANGLE Cipher

Input	Output
$1_x$	??1?
$4_x$	??11
$5_x$	??0?
$8_x$	???1
$C_x$	???0

turbed bits are used for different attacks of PRESENT [64] and SERPENT [69] block ciphers.

These bits can be used to find longer differential characteristics leading to more effective differential attack. In [54], it was shown that the attack on PRESENT cipher reduces to 7 rounds when the S-box is replaced with a similar one that lacks undisturbed bits.

#### 4.2.8 Differential Factors

In a differential attack, all possible keys are tried on plain text pairs that are waiting to satisfy different characteristics that enable us to guess the right subkeys. In order to distinguish the correct key from the wrong key, the correct key must satisfy the differential characteristic more than any other key. However, in some cases, the output difference of the S-box process may be invariant when the round key is XORed with a certain value. For this reason, some candidate keys may satisfy differential characteristics for an equal number of times. In such case, the attacker can not capture the whole round key bits. The Differential Factors were first proposed in [58] as follows:

**Definition 4.2.6** *Let  $S$  be a function from  $F_2^n$  to  $F_2^m$ . For all  $x, y \in F_2^n$  that satisfy  $S(x) \oplus S(y) = \mu$ , if we also have  $S(x \oplus \lambda) \oplus S(y \oplus \lambda) = \mu$ , then we say that the S-box has a differential factor  $\lambda$  for the output difference  $\mu$ . (i.e.  $\mu$  remains invariant for  $\lambda$ )*

Differential factors are observed mostly in small S-boxes of cryptographic algorithms as discussed in [58]. Since lightweight block ciphers use small S-boxes to provide trade-off between power and security, most of them have differential factors in their S-boxes. For example, SERPENT and PRESENT are two different lightweight block ci-

phers that contain several differential factors. As shown in [58], [55] SERPENT has 8 differential factors in their 4 S-boxes, PRESENT has 2 differential factors. With the help of these factors, the differential attack [7] of SERPENT can be performed with the time complexities reduced by a factor of 4. In addition, the differential attack [62] of PRESENT actually requires  $2^{70}$  memory accesses instead of  $2^{64}$  with them.

In [58], it is proven that the number of differential factors of a S-box is the same with the number of differential factors of its inverse. Moreover, it also provides the differential factors of the inverse S-box when we know the differential factors of the S-box. Hence, there is no need to check the differential factors of the inverse of S-boxes.

**Theorem 4.2.7** *If a bijective S-box  $S$  has a differential factor  $\lambda$  for an output difference  $\mu$ , then  $S^{-1}$  has a differential factor  $\mu$  for the output difference  $\lambda$ .*

We have used the theorem above in our study because the S-box of RECTANGLE block cipher was changed with the inverse of it to pretend differential attacks.

**Theorem 4.2.8** [58] *If  $\lambda_1$  and  $\lambda_2$  are differential factors for an output difference  $\mu$ , then  $\lambda_1 \oplus \lambda_2$  is also a differential factor for the output difference  $\mu$ . i.e. All differential factors  $\lambda_i$  for  $\mu$  form a vector space.*

### 4.3 Differential Factors and Cryptanalysis

If a block cipher contains key XOR process in front of the substitution process, then differential factor may affect cryptanalysis. In SPN type block ciphers, we could face with this structure mostly; but this is not the case every time. Some conditions must be satisfied;

1. There must be a differential factor  $\lambda$  for output difference  $\mu$  for a S-box activated by the attack.
2. The differential being used in the attack requires the output difference of this S-box to be  $\mu$ .

In differential cryptanalysis, the correct key is discovered by trying all candidate keys. If a key  $k$  is the correct key then it must satisfy differential characteristic more times than any other key. However, if S-box has differential factor, in a differential attack for any key  $k$ ,  $k$  and  $k \oplus \lambda$  would get the same number of hits since  $\lambda$  is a differential factor. Hence the attacker cannot distinguish half of the guessed keys with the other half. As a result, during key guess step it is not possible to discover the key bits where differential factors exist. These key bits must be discovered with the exhaustive search. This decreases the time complexity of the key guess step. Because additional bits have to be discovered, time complexity of exhaustive search step is increased. This theorem is proved in [58].

**Theorem 4.3.1** [58] *In a block cipher let an S-box  $S$  contain a differential factor  $\lambda$  for an output difference  $\mu$  and the partial round key  $k$  is XORed with the input of  $S$ . If an input pair provides the output difference  $\mu$  under a partial subkey  $k$ , then the same output difference is observed under the partial subkey  $k \oplus \lambda$ . Therefore, during a differential attack involving the guess of a partial subkey corresponding to the output difference  $\mu$ , the advantage of the cryptanalyst is reduced by 1 bit and the time complexity of this key guess step is halved.*

**Corollary 4.1** [58] *During a differential attack involving the guess of a partial subkey corresponding to the output difference  $\mu$  of an S-box that has a vector space of differential factors of dimension  $r$  for  $\mu$ , the advantage of the cryptanalyst is reduced by  $r$  bits and the time complexity of the key guess step is reduced by a factor of  $2^r$ .*

**Corollary 4.2** *Differential factors reduce the key space for the key guess process and therefore reduce the data complexity of the attack. Thus, memory required to keep the counters for the guessed keys also reduces. Reduction in the data complexity may also reduce the time complexity depending on the attack.*

During our research, we have discovered the existence of differential factors on RECTANGLE. The differential factors are listed in Table 6.5.

## CHAPTER 5

### ATTACKS ON BLOCK CIPHERS

Up to this point, we have analyzed the structure of block ciphers. In this part, we investigate the attacks on block ciphers; so that we can understand better why the block ciphers are in this structure.

**Kerchoff's Principle:** In 1883 Auguste Kerckhoffs wrote two journal articles on *La Cryptographie Militaire* that includes the following principles. They gave a practical, experience-based approach, including six design principles for military ciphers:

1. The system must be practically, if not mathematically, indecipherable.
2. It must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience.
3. Its key must be communicable and retainable without the help of written notes.
4. It must be applicable to telegraphic correspondence.
5. Apparatus and documents must be portable, and its usage and function must not require the concurrence of several people.
6. Finally, it is necessary, given the circumstances that command its application, that the system be easy to use, requiring neither mental strain nor the knowledge of a long series of rules to observe.

Nowadays, the second principle is widely known and only it is referred as Kerchoff's Principle by the most people which is a common mistake. This principle suggests that cryptosystem should be secure even if everything about the system, except the

key, is publicly known. In other words, the encryption algorithm should not be secret and the security should be provided by keeping the key as secret in cryptographic systems. To obtain the secret key, the most obvious method is trying every possible key to decrypt the ciphertext. This method is also known as exhaustive search or brute force attack. This attack is performed by obtaining plaintext-ciphertext pairs. By encrypting plaintexts with every possible key you can find the correct key when the cipher text matches with the previously obtained ciphertext. Since this attack type is very simple, it can be used for all block ciphers. Against to brute force attacks, the key length should be selected as long as possible considering the power of the current technology. Another method is table attack; every possible corresponding plaintext - ciphertext pairs for encrypting key are obtained and stored in a database. Then decrypting a cipher text requires only a database query operation that finds matching plaintext for a ciphertext. In this attack,  $2^n$  data must be stored for the n-bit block length. If an attack that captures encrypting key for a block cipher with less operations than exhaustive search and needed less data than table attack, then that block cipher is considered as broken. The following terminology is also useful to know:

**Corollary 5.1** *An encryption scheme is called computationally secure if:*

1. *The cost of breaking the cipher exceeds the value of the encrypted information*
2. *The time required to break the cipher exceeds the useful lifetime of the information.*

Attacks to block ciphers can be categorized according to the information that is required to perform the attack:

- 1) **Ciphertext-only attack:** In this attack the attacker knows only the ciphertext information. The attacker will try to find the key or decrypt one or more pieces of ciphertext. The cipher should have significant weaknesses to perform a cipher text-only attack (e.g. A5/1 Stream Cipher)
- 2) **Known-plaintext attack:** In this attack, attackers can get n plaintexts and the corresponding ciphertexts. Exhaustive search or table attacks can be examples

of known-plaintext attack.

- 3) **Chosen-plaintext attack:** This is a known plaintext attack in which the attacker can choose the plaintext to be encrypted and read the corresponding ciphertext.
- 4) **Chosen-ciphertext attack:** The attacker has the ability to select any ciphertext and study the plaintext produced by decrypting them.
- 5) **Adaptive chosen-plaintext (ciphertext) attack:** In this attack, the attacker is able to request encryptions of some plaintexts possibly seeing them first and then making some calculations using them. Collecting data becomes harder as we move down the list.
- 6) **Chosen-key attack:** In this attack, the attacker is able to choose the relation between two keys, used to encrypted two sets of plaintexts.

## 5.1 Cryptanalytic Attacks

Cryptanalysis is used to break cryptographic security systems and gain access to the secret messages, even if the cryptographic key is unknown. Differential cryptanalysis is one of the most important techniques investigate the security of a block cipher.

### 5.1.1 Differential Cryptanalysis

Although differential cryptanalysis have been discovered at least 30 years ago it was not reported in the open literature until 1990. The first published effort appears in late 1980s by a number of papers by Biham and Shamir. There are several types of differential cryptanalysis. These are:

#### 1) Related-Key Differential Cryptanalysis

The related key attack model [4] is the class of cryptanalytic attacks which the attacker knows or selects the relationship between several keys and he can encrypt / decrypt plaintexts with all these keys. The main aim of the attacker

is finding actual secret keys. The relation between the keys can be an arbitrary bijective function. This relation is provided by just XOR with a constant.

$K_2 = K_1 \oplus C$ , where  $C$  can be chosen by attacker. Such a type relation allows the attacker to monitor the propagation of XOR differences that are induced by key difference  $C$  via the key-schedule of the cipher. This is the most complex form of related-key differential attack. In the more complex forms, attacker can associate other relations among the keys. For example in [8] the attacker chooses a desired XOR relation in the second subkey. Then he defines the new relation between actual keys as following:

$$K_2 = F^{-1}(F(K_1) \oplus C) = RC(K_1),$$

where  $F$  is round function of AES-256 key schedule and  $C$  is the chosen constant.

In Chapter 4, we have investigated related-key differential attack on RECTANGLE. AES [8] is the another cipher which is applied related-key differential cryptanalysis.

## 2) Truncated Differential Cryptanalysis [30]

Only some part of difference in a pair can be guessed by truncated differential cryptanalysis after each round. More than one truncated differential characteristics can be used together to reduce the time complexity of the attack. CRYPTON [27], PRINCE [70], SKIPJACK[41], SALSA20 [16] are block ciphers which are applied truncated differential cryptanalysis.

## 3) Higher Order Differential Cryptanalysis [65]

While differential cryptanalysis explores the difference between two inputs, higher-order differential cryptanalysis explores the effects of some differences between a wider set of inputs. SPECTR-H64 [32] is one of the cipher which is applied higher-order differential cryptanalysis.

## 4) Impossible Differential Cryptanalysis [6]

Whereas regular differential cryptanalysis searches for differences higher than expected probability, impossible differential cryptanalysis investigates differences that are impossible. Since the probability of such differences is zero, any



candidate key that is tried in the cryptanalysis process can not be the correct key. AFRICACRYPT [60], CAMELLIA [26], MIDORI [11], CLEFIA [48] are the ciphers applied impossible differential cryptanalysis.

**5) Improbable Differential Cryptanalysis** This cryptanalysis technique examines the differences that are less likely to exist for the correct key than a wrong key. PRESENT [53] and CLEFIA [59] are the ciphers which are applied improbable differential cryptanalysis.

#### 5.1.1.1 Overview of Basic Attack

Differential cryptanalysis is a chosen plaintext attack and it investigates the relations of input differences and the corresponding output differences. Let consider a system with input  $X = [X_1 X_2 \dots X_n]$  and output  $Y = [Y_1 Y_2 \dots Y_n]$ . Let two inputs to the system be  $X'$  and  $X''$  with the corresponding outputs  $Y'$  and  $Y''$ , respectively. The input difference is given by  $\Delta X = X' \oplus X''$  and corresponding output difference is  $\Delta Y = Y' \oplus Y''$ .

In a ideal cipher, a particular output difference  $\Delta Y$  occurs given a particular input difference  $\Delta X$  with probability  $\frac{1}{2^n}$  where  $n$  is the block length. The main aim of differential cryptanalysis is finding a scenario where a particular  $\Delta Y$  occurs given a particular input difference  $\Delta X$  with the probability higher than  $\frac{1}{2^n}$ . The pair  $(\Delta X, \Delta Y)$  is called **differential**. If after  $r$  rounds,  $\Delta X$  input difference causes  $\Delta Y$  output difference with some probability, it is called as **differential characteristic**. A differential characteristic can be used to distinguish random encryption from the specific block cipher by encrypting  $N$  plaintext with fixed key and comparing the  $\Delta X$  and  $\Delta Y$  results with the differential characteristic. We expect to observe differential characteristic in  $N$  plain text encryption with an expected value which is calculated by multiplying differential characteristic probability and number of pairs.

#### 5.1.1.2 Extracting Key Bits

To extract some key bits, one or more rounds are added to behind or front of the  $r$  round differential. For example, if we have 4 rounds differential characteristics such

that  $\Delta X = 01000000$  goes to  $\Delta Y = 00700000$ . We can add one round encryption above the 4 round differential, then we can investigate the DDT of the related S-boxes to find out possible input difference values that causes to  $\Delta X = 01000000$ . For this input difference, we can pick random input values and encrypt them with every possible key. After that encryption, if the output difference  $\Delta Y = 00700000$ , then we can increase the counter value of the key. The correct key have the greatest counter value. In this way, we can extract some bits of the subkeys.

### 5.1.1.3 Complexity of the Attack

In cryptanalysis, there are three things that dictate the complexity of the attack. The first is **Data Complexity** that need to be acquired by an attacker. The second is **Time Complexity** of the analysis required to derive the secret key. The third is **Memory Complexity**, the amount of storage required to perform the attack.

## 5.1.2 An Example of Differential Cryptanalysis: Differential Cryptanalysis of UltraLightweight PRESENT Cipher

The best known differential attack on PRESENT is provided in [62] by adding two rounds to the bottom of the 24 different 14-round differentials which has different input and same output difference. Recently, it was shown in [55] that this attack overlooks 6 differential factors and therefore the number of bits that are actually captured is 6 fewer than what is claimed. In this work we give another correction to this attack due to undisturbed bits.

The differential distribution table is as shown in Table 5.1. If one examines that table, he can easily notices that the differential uniformity of PRESENT's S-box is 4.

PRESENT's S-box has also following differential factors: The best known differential attack on PRESENT [62] breaks 16 round by adding two rounds to the bottom of 14 round differential  $\Delta_1$  which has a probability of  $2^{-62}$ .

$$\Delta_1: 07000000000000700 \rightarrow 0000000900000009$$

Table 5.1: Differential distributions of the S-box in PRESENT

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	4	0	0	0	4	0	4	0	0	0	4	0	0
2	0	0	0	2	0	4	2	0	0	0	2	0	2	2	2	0
3	0	2	0	2	2	0	4	2	0	0	2	2	0	0	0	0
4	0	0	0	0	0	4	2	2	0	2	2	0	2	0	2	0
5	0	2	0	0	2	0	0	0	0	2	2	2	4	2	0	0
6	0	0	2	0	0	0	2	0	2	0	0	4	2	0	0	4
7	0	4	2	0	0	0	2	0	2	0	0	0	2	0	0	4
8	0	0	0	2	0	0	0	2	0	2	0	4	0	2	0	4
9	0	0	2	0	4	0	2	0	2	0	0	0	2	0	4	0
A	0	0	2	2	0	4	0	0	2	0	2	0	0	2	2	0
B	0	2	0	0	2	0	0	0	4	2	2	2	0	2	0	0
C	0	0	2	0	0	4	0	2	2	2	2	0	0	0	2	0
D	0	2	4	2	2	0	0	2	0	0	2	2	0	0	0	0
E	0	0	2	2	0	0	2	2	2	2	0	0	2	2	0	0
F	0	4	0	0	4	0	0	0	0	0	0	0	0	0	4	4

Table 5.2: Differential Factors of PRESENT

<b>S-box</b>	0123456789ABCDEF	$\lambda$	$\mu$
PRESENT	C56B90AD3EF84712	1	5
PRESENT	C56B90AD3EF84712	$F$	$F$

Authors claim to capture 32 bits with  $2^{33.18}$  2-round encryptions, and remaining 48 bits are captured with  $2^{48}$  16-round encryptions via exhaustive search. However this attack is corrected in [55] that authors fail to discover 6 differential factors which are shown in Table 5.3. Therefore, as explained in [55], the number of bits that are actually captured is 26 bits not 32 bits which require  $2^{27.18}$  2-round encryptions and remaining 54 bits require  $2^{54}$  16-round encryptions.

The output difference of the characteristics activates the S-boxes  $S_0$  and  $S_8$  in the round 15 and  $S_4, S_6, S_8, S_{10}, S_{12},$  and  $S_{14}$  in the round 16 which is shown in Table 5.3. Thus, this differential attack captures 32 bits of the key with a time complexity of  $2^{33.18}$  2-round PRESENT encryptions, a data complexity of  $2^{64}$  chosen plaintexts, and a memory complexity of  $2^{32}$  6-bit counters. This part of the attack works with a success probability of 99.999993% and then the remaining 48 bits are obtained via exhaustive search which requires  $2^{48}$  16-round PRESENT encryptions.

Table 5.3: 16-round differential-linear attack of [62]. Values that need to be obtained are shown in bold.

Rounds	Differences in bits															
	$x_{15}$	$x_{14}$	$x_{13}$	$x_{12}$	$x_{11}$	$x_{10}$	$x_9$	$x_8$	$x_7$	$x_6$	$x_5$	$x_4$	$x_3$	$x_2$	$x_1$	$x_0$
$X_{1,I}$	0000	0111	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0111	0000	0000
14-Round Differential $\Delta_1$																
$X_{14,P}$	0000	0000	0000	0000	0000	0000	0000	<b>1001</b>	0000	0000	0000	0000	0000	0000	0000	<b>1001</b>
$X_{15,S}$	0000	0000	0000	0000	0000	0000	0000	<b>??0</b>	0000	0000	0000	0000	0000	0000	0000	<b>??0</b>
$X_{15,P}$	0000	000?	0000	000?	0000	000?	0000	000?	0000	000?	0000	000?	0000	0000	0000	0000
$X_{16,S}$	0000	????	0000	????	0000	????	0000	????	0000	????	0000	????	0000	<b>0000</b>	0000	<b>0000</b>

## 5.2 Corrected Attack on PRESENT

The activated S-boxes of the round 16 have the input difference 1 and inverse of PRESENT's S-box has a differential factor  $\lambda = 5$  for  $\mu = 1$ . Thus,  $\mu = 1$  coincides with the input difference of these six S-boxes and it was shown in [55] that the advantage of this attack is actually 26 bits instead of 32 bits. This theoretical result is also experimentally verified by removing the first few rounds of the 14-round differential so that it remains within our computational power. This observation reduces the time complexity of the first part of the attack to  $2^{27.18}$  2-round PRESENT encryptions and the memory complexity to  $2^{26}$  6-bit counters. However, the time complexity of ex-

haustive search for the remaining bits of the key is  $2^{54}$  16-round PRESENT encryptions, instead of  $2^{48}$  as it was claimed. We further give a correction to this attack due to the undisturbed bits. Since the input difference 9 for the S-box only activates the most significant three bits, it was assumed that we need to capture the values of three S-boxes in the 16<sup>th</sup> round. However, we cannot verify the characteristic without knowing the all four bits of the S-box output in the 15<sup>th</sup> round. We provided the parts that need to be obtained in bold in Table 5.3. Thus, the attacker also needs to guess the 16<sup>th</sup> round subkeys corresponding to  $S_0$  and  $S_2$ . But the attacker's advantage increases by 6 instead of 8 bits due to the following property.

**Property 5.1 ([57])** *Inverse of PRESENT's S-box  $S$  has the property  $lsb(S^{-1}(x)) = lsb(S^{-1}(x \oplus 5))$  where  $lsb$  is the least significant bit.*

Thus, a correct differential attack on 16-round PRESENT needs to guess 32 key bits in the 16-th round that correspond to the nibbles  $x_0, x_2, x_4, x_6, x_8, x_{10}, x_{12}, x_{14}$  and 8 key bits in the 15-th round. However, this attack provides 32-bit advantage to the attacker instead of 40 bits because of the 6 differential factors corresponding to the nibbles  $x_4, x_6, x_8, x_{10}, x_{12}, x_{14}$  and the application of Property 5.1 to the nibbles  $x_0$  and  $x_2$ . Thus, the whole 80-bit key can be obtained after an exhaustive search that requires  $2^{48}$  16-round PRESENT encryptions.



## CHAPTER 6

### OVERVIEW OF RECTANGLE

Small embedded devices such as sensor nodes, smart cards and RFIDs are widely used nowadays. This widespread use has also brought security problems. Thus, symmetric-key ciphers, especially block ciphers have gain importance in the security of small embedded devices since they provide strong security at a lower cost. RECTANGLE is one of the lightweight block cipher designed to meet this need.

Since bit-sliced technique is used in the design of RECTANGLE algorithm, it achieves a very low cost in hardware and a very competitive performance in software. The block length is 64 bits and two key lengths of 80 and 128 bits are supported.

It is a SPN type cipher. While 16  $4 \times 4$  S-boxes are used in the substitution layer, permutation is provided by using three rotations in the permutation layer. Before discussing these layers in details, it is useful to talk about 3 main advantages of RECTANGLE.[69]

1. Because RECTANGLE is designed according to bit-sliced design principle, it is extremely suitable for hardware; it allows very efficient and flexible hardware implementations. These data obtained in [69] can be helpful to explain these arguments: with 80-bit key version, by using  $0.13\mu\text{m}$  standard cell library at 100 KHz, a throughput of 246 Kbits/sec and an energy efficiency of 3.0 pJ/bit with only 1600 gates was obtained for round-based implementation, a throughput of 14.0 Kbits/sec and an energy efficiency of 32.05 pJ/bit with only 1111 gates was obtained for serialized implementation.
2. Again due to bit-slice style, RECTANGLE is faster than most of the other light-

weight block ciphers. Easy implementation of S-box and simple rotation operations in permutation layer make RECTANGLE very friendly for both hardware and software. In [69], this collaboration can be exemplified with these data; bit-slice implementation is achieved by a speed of about 30.5 cycles/byte for encryption and 32.2 cycles/byte for decryption for one block of data. Implementation with a parallel mode of operation of RECTANGLE reaches speed of about 3.9 cycles/byte for messages around 3000 bytes for encryption by using Intel 128-bit SSE instructions on a 2.5GHz Intel(R) Core i5 – 2520M CPU.

3. The trade-off between security and performance is also achieved by RECTANGLE with careful selection of S-box and asymmetric design of rotations in permutation layer.

## 6.1 The Last Version of The RECTANGLE Algorithm

After introducing RECTANGLE algorithm in 2014 in [69], researchers from Chinese Academy of Sciences found all 15-round differential characteristics with 26 – 30 active S-boxes for given input, output and round subkey differences, which have a total probability of  $2^{-60.5}$ . After this attempt, designers of RECTANGLE have changed its key schedule and S-box. In this thesis, we have mentioned about the old type of RECTANGLE as REC-0 which we introduce you in details in the next section.

### 6.1.1 Representation of Plaintext and Subkey as Matrix

A 64-bit plaintext or a 64-bit intermediate result or a 64-bit ciphertext is called cipher state and represented as a  $4 \times 16$  rectangular array of bits. If  $P = p_{63}||\dots||p_1||p_0$  denotes cipher state, then we can arrange rows as in figure;

$$P = \begin{bmatrix} p_{15} & \dots & p_2 & p_1 & p_0 \\ p_{31} & \dots & p_{18} & p_{17} & p_{16} \\ p_{31} & \dots & p_{34} & p_{33} & p_{32} \\ p_{63} & \dots & p_{50} & p_{49} & p_{48} \end{bmatrix} P = \begin{bmatrix} p_{0,15} & \dots & p_{0,2} & p_{0,1} & p_{0,0} \\ p_{1,15} & \dots & p_{1,2} & p_{1,1} & p_{1,0} \\ p_{2,15} & \dots & p_{2,2} & p_{2,1} & p_{2,0} \\ p_{3,15} & \dots & p_{3,2} & p_{3,1} & p_{3,0} \end{bmatrix}$$



## 6.1.2 Substitution and Permutation Operations

RECTANGLE algorithm consists of 25 rounds; in each round there are 3 main operations; adding subkey, substituting column and shifting rows. Adding subkey is simple XOR operation of cipher state and subkey matrices. In substitution operation, the S-box is applied to each column of the cipher state in parallel as the following way;

$$\begin{bmatrix} p_{0,15} \dots p_{0,2} p_{0,1} p_{0,0} \\ p_{1,15} \dots p_{1,2} p_{1,1} p_{1,0} \\ p_{2,15} \dots p_{2,2} p_{2,1} p_{2,0} \\ p_{3,15} \dots p_{3,2} p_{3,1} p_{3,0} \end{bmatrix} \rightarrow \begin{bmatrix} S(p_{0,15}) \dots S(p_{0,2}) S(p_{0,1}) S(p_{0,0}) \\ S(p_{1,15}) \dots S(p_{1,2}) S(p_{1,1}) S(p_{1,0}) \\ S(p_{2,15}) \dots S(p_{2,2}) S(p_{2,1}) S(p_{2,0}) \\ S(p_{3,15}) \dots S(p_{3,2}) S(p_{3,1}) S(p_{3,0}) \end{bmatrix}$$

In the ShiftRow step, the last three rows are left rotated 1, 12, and 13 bits, respectively as in figure. After 25 rounds of iterations, there is a final subkey XOR.

$$\begin{aligned} (p_{15} \dots p_2 p_1 p_0) &\xrightarrow{\lll 0} (p_{15} \dots p_2 p_1 p_0) \\ (p_{31} \dots p_{18} p_{17} p_{16}) &\xrightarrow{\lll 1} (p_{30} \dots p_{17} p_{16} p_{31}) \\ (p_{47} \dots p_{34} p_{33} p_{32}) &\xrightarrow{\lll 12} (p_{35} \dots p_{38} p_{37} p_{36}) \\ (p_{63} \dots p_{50} p_{49} p_{48}) &\xrightarrow{\lll 13} (p_{50} \dots p_{53} p_{52} p_{51}) \end{aligned}$$

### 6.1.2.1 S-box

The S-box used in RECTANGLE is as given by the Table 6.1 ;

Table 6.1: S-box of RECTANGLE

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S(x)	6	5	C	A	1	E	7	9	B	0	3	D	8	F	4	2

### 6.1.2.2 Differential Factors

Differential factors of S-box used in RECTANGLE as shown in Table 6.2;

Table 6.2: Differential Factors of RECTANGLE

S-box	0123456789ABCDEF	$\lambda$	$\mu$
RECTANGLE	65CA1E79B03D8F42	4	2
RECTANGLE	65CA1E79B03D8F42	C	E

### 6.1.3 Key Schedule

As a key, RECTANGLE accepts 80 or 128 bits keys. Firstly, we introduce the usage of the 80-bits key. When the key is provided by a user as 80 bits like  $V = v_{79} \parallel \dots \parallel v_1 \parallel v_0$ , the key is stored in an 80 bits key register and arranged as  $5 \times 16$  array of bits as shown in below.

$$\begin{bmatrix} v_{15} & \dots & v_1 & v_0 \\ v_{31} & \dots & v_{17} & v_{16} \\ v_{47} & \dots & v_{33} & v_{32} \\ v_{63} & \dots & v_{49} & v_{48} \\ v_{79} & \dots & v_{65} & v_{64} \end{bmatrix} \begin{bmatrix} K_{(0,15)} & \dots & K_{(0,1)} & K_{(0,0)} \\ K_{(1,15)} & \dots & K_{(1,1)} & K_{(1,0)} \\ K_{(2,15)} & \dots & K_{(2,1)} & K_{(2,0)} \\ K_{(3,15)} & \dots & K_{(3,1)} & K_{(3,0)} \\ K_{(4,15)} & \dots & K_{(4,1)} & K_{(4,0)} \end{bmatrix}$$

In key register,  $Row_i = K_{(i,15)} \& \dots \& K_{(i,1)} \& K_{(i,0)}$  for  $0 \leq i \leq 4$  denotes 16-bits word. In each round, the first 4 rows of the current contents of the key register is selected to be XORed with 64-bits cipher state; i.e.,  $K_i = Row_3 \parallel Row_2 \parallel Row_1 \parallel Row_0$ . After completing selection part of  $K_i$ , the key is updated as follows:

1. S-box is applied to the the bits intersected at the 4 uppermost rows and the 4 rightmost columns like as follows:

$$K'_{3,j} \parallel K'_{2,j} \parallel K'_{1,j} \parallel K'_{0,j} := S(K_{3,j} \parallel K_{2,j} \parallel K_{1,j} \parallel K_{0,j}), j= 0, 1, 2, 3$$

2. A 1-round generalized Feistel transformation is applied as follows:

$$Row'_0 := (Row_0 \lll 8) \oplus Row_1,$$

$$Row'_1 := Row_2,$$

$$Row'_2 := Row_3,$$

$$4Row'_3 := (Row_3 \lll 12) \oplus Row_4,$$

$$Row'_4 := Row_0$$

3. A 5-bit round constant  $RC[i]$  is XORed with the 5-bit key state  $(K_{0,4} \parallel K_{0,3} \parallel K_{0,2} \parallel K_{0,1} \parallel K_{0,0})$ , i.e.,

$$K'_{(0,4)} \parallel K'_{(0,3)} \parallel K'_{(0,2)} \parallel K'_{(0,1)} \parallel K'_{(0,0)} = K_{(0,4)} \parallel K_{(0,3)} \parallel K_{(0,2)} \parallel K_{(0,1)} \parallel K_{(0,0)} \oplus RC[i]$$

At the end,  $K_{25}$  is extracted from the updated key state. All round constants  $RC[i]$  ( $i = 0, 1, \dots, 24$ ) are generated by a 5-bit LFSR. At each round, the 5 bits  $(rc_4, rc_3, rc_2, rc_1, rc_0)$  are left shifted over 1 bit, with the new value of  $rc_0$  being computed as  $rc_4 \oplus rc_2$ . The initial value is defined as  $RC[0] := 0x1$ . The round constants are listed in Table 6.3.

Table 6.3: Round Constants of Rectangle Cipher

RC[0] = 0X01	RC[5] = 0X05	RC[10] = 0X13	RC[15] = 0X1C	RC[20] = 0X0D
RC[1] = 0X02	RC[6] = 0X0B	RC[11] = 0X07	RC[16] = 0X18	RC[21] = 0X1B
RC[2] = 0X04	RC[7] = 0X16	RC[12] = 0X0F	RC[17] = 0X11	RC[22] = 0X17
RC[3] = 0X09	RC[8] = 0X0C	RC[13] = 0X1F	RC[18] = 0X03	RC[23] = 0X0E
RC[4] = 0X12	RC[9] = 0X19	RC[14] = 0X1E	RC[19] = 0X06	RC[24] = 0X1D

When the key is provided by user as 128 bits, the key is stored in a 128 bits key register and arranged as  $4 \times 32$  array of bits as shown in below.

$$\begin{bmatrix} K_{(0,31)} & \dots & K_{(0,2)} & K_{(0,1)} & K_{(0,0)} \\ K_{(1,31)} & \dots & K_{(1,2)} & K_{(1,1)} & K_{(1,0)} \\ K_{(2,31)} & \dots & K_{(2,2)} & K_{(2,1)} & K_{(2,0)} \\ K_{(3,31)} & \dots & K_{(3,2)} & K_{(3,1)} & K_{(3,0)} \end{bmatrix}$$

In key register,  $Row_i = K_{(i,31)} \& \dots \& K_{(i,1)} \& K_{(i,0)}$  for  $0 \leq i \leq 3$  denotes 32-bits word.

In each round, the 64-bit round subkey  $K_i$  consists of the 16 rightmost columns of

the current contents of the key. After completing selection part of  $K_i$ , the key is updated as follows:

1. S-box applies to the bits intersected at the 4 uppermost rows and the 4 rightmost columns as follows:

$$K'_{3,j} \parallel K'_{2,j} \parallel K'_{1,j} \parallel K'_{0,j} := S(K_{3,j} \parallel K_{2,j} \parallel K_{1,j} \parallel K_{0,j}), 0 \leq j \leq 7$$

2. A 1-round generalized Feistel transformation is applied as follows:

$$Row'_0 := (Row_0 \lll 8) \oplus Row_1,$$

$$Row'_1 := Row_2,$$

$$Row'_2 := (Row_2 \lll 16) \oplus Row_3,$$

$$Row'_3 := Row_0$$

3. A 5-bit round constant  $RC[i]$  is XORed with the 5-bit key state ( $K_{0,4} \parallel K_{0,3} \parallel K_{0,2} \parallel K_{0,1} \parallel K_{0,0}$ ), where  $RC[i], (i=0,1,\dots,24)$  are the same as those used in the 80-bit key schedule.

At the end,  $K_{25}$  is extracted from the updated key state. All round constants  $RC[i]$  ( $i=0,1,\dots,24$ ) are generated by a 5-bit LFSR. At each round, the 5 bits ( $rc_4, rc_3, rc_2, rc_1, rc_0$ ) are left shifted over 1 bit, with the new value of  $rc_0$  being computed as  $rc_4 \oplus rc_2$ . The initial value is defined as  $RC[0] := 0x1$ . We list all the round constants in Table 6.3.

#### 6.1.4 Whole Cipher

The RECTANGLE Algorithm consists of 25 rounds and a final subkey XOR.

A pseudo code is as follows:

```

GenerateRoundKeys()
for i = 0 to 24 do
{
    AddRoundKey(STATE, Ki )
    SubColumn(STATE)
    ShiftRow(STATE)
}
AddRoundKey(STATE, K25 )

```

## 6.2 The REC-0(First Version of the RECTANGLE Algorithm)

After the article that contains the 19-round differential attack was published, the algorithm was changed to provide better security. The initial design of RECTANGLE, which is now referred to as REC-0, has a different key schedule and uses the inverse of RECTANGLE's S-box. Now we inform you about this design.

In this design, as in the last version of RECTANGLE there are 25 rounds. Each round is composed of three steps: AddRoundkey, SubColumn and ShiftRow.

### 6.2.1 Representation of Plaintext and Subkey as Matrix

A 64-bit plaintext, or a 64-bit intermediate result, or a 64-bit ciphertext is called cipher state and represented as a  $4 \times 16$  rectangular array of bits. If  $P = p_{63}||\dots||p_1||p_0$  denotes cipher state, we can arrange rows as in figure;

$$P = \begin{bmatrix} p_{15} & \dots & p_2 & p_1 & p_0 \\ p_{31} & \dots & p_{18} & p_{17} & p_{16} \\ p_{31} & \dots & p_{34} & p_{33} & p_{32} \\ p_{63} & \dots & p_{50} & p_{49} & p_{48} \end{bmatrix}$$

### 6.2.2 Substitution and Permutation Operations

In each round of RECTANGLE algorithm, there are 3 main operations; adding subkey, substituting column and shifting rows. Adding subkey is simple XOR operation of cipher state and subkey matrices, in substitution operation, the S-box is applied to each column of the cipher state in parallel as in RECTANGLE;

In the ShiftRow step, the last three rows are left rotated 1, 12, and 13 bits, respectively as in described above in RECTANGLE. After 25 rounds of iterations, there is a final subkey XOR operation left.

#### 6.2.2.1 S-box

The S-box used in REC-0 is the inverse of the last version's S-box as you can examine in Table 6.4;

Table 6.4: S-box of REC-0

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S(x)	9	4	F	A	E	1	0	6	C	7	3	8	2	B	5	D

#### 6.2.2.2 Differential Factors

Differential factors is also inverse of the last version's differential factors as we figured out in Theorem 4.2.7. You can examine in Table 6.5.

Table 6.5: Differential Factors of the S-box of REC-0

S-box	0123456789ABCDEF	$\lambda$	$\mu$
REC-0	94FAE106C7382B5D	2	4
REC-0	94FAE106C7382B5D	E	C

### 6.2.3 Key Schedule

The key schedule of RECTANGLE is composed of three steps. The S-box is the same as in a round transformation. The key arranged as a  $4 \times 20$  array of bits like in figure:

$$P = \begin{bmatrix} k_{(0,20)} & \dots & k_{(0,2)} & k_{(0,1)} & k_{(0,0)} \\ k_{(1,20)} & \dots & k_{(1,2)} & k_{(1,1)} & k_{(1,0)} \\ k_{(2,20)} & \dots & k_{(2,2)} & k_{(2,1)} & k_{(2,0)} \\ k_{(3,20)} & \dots & k_{(3,2)} & k_{(3,1)} & k_{(3,0)} \end{bmatrix}$$

The rows are left rotated 7 bits, 9 bits, 11 bits and 13 bits, respectively. The round constants  $RC[i]$  ( $0 \leq i \leq 24$ ) are generated by a 5-bit LFSR with the initial state  $RC[0]=(0,0,0,0,1)$ . At each round  $i \geq 0$ , the round constant  $RC[i] = (r_{i,4}, \dots, r_{i,1}, r_{i,0})$  is equal to  $(r_{i-1,3}, \dots, r_{i-1,0}, r_{i-1,4} \oplus r_{i-1,2})$ .

### 6.2.4 Whole Cipher

The pseudocode of the REC-0 is the same as in the last version of RECTANGLE algorithm as we mentioned above.

## 6.3 Differential Attacks on RECTANGLE

Before introducing differential attacks on RECTANGLE, some notations used in attacks are as follows;

- $P$  and  $(P')$  denote plaintext pairs,
- $C$  and  $(C')$  denote ciphertext pairs,
- $\Delta P$  and  $\Delta C$  denote differences of the plaintext and ciphertext,
- $K_i, (K_i^i)$  denote the round subkey,
- $I_i, (I_i^i)$  denote the input of the operation SubColumn in the  $i^{th}$  round,

- $O_i, (O_i^i)$  denote the input of the operation SubColumn in the  $i^{th}$  round,
- $\Delta K_i$  denotes the difference of the round subkey,
- $\Delta I_i$  denotes the difference of the input of the operation SubCoulmn in the  $i^{th}$  round,
- $\Delta O_i$  denotes the difference of the input of the operation SubCoulmn in the  $i^{th}$  round.

After mentioning preparations above, we introduce you about related key differential attack on 19-round REC-0. In this attack, there is a 15-round differential characteristics from second round to seventeenth round with the same input, output and round subkey differences. A related-key differential attack on 19-round REC-0 is obtained by extending 15 round differential characteristics to 2 round backward and forward.

### 6.3.1 19-round Related-key Differential Attack on REC-0

#### 6.3.1.1 Differential Characteristics

According to Difference Distiribution Table of the S-box in RECTANGLE below, we can compute the probability for a specific differential characteristic. The results of related related-key attacks on the n-round reduced REC-0 are showed in Table 6.6., where  $7 \leq n \leq 15$ . The values of the 'Probability' column are the probabilities of specific differential characteristics which corresponds to the numbers of active S-boxes in Table 6.7. When the numbers of the rounds are 7, 8 and 9 marked by ' \* ', the values in the column 'Number of active S-boxes' correspond to the minimal numbers of active S-boxes for all possible differential characteristics. When the numbers of the rounds are 10, 11, 12, 13, 14, 15, the values in the column 'Number of active S-boxes' correspond to the minimal numbers of active S-boxes for some possible differential characteristics.

In [45], a 15-round differential characteristic with a probability  $2^{-64}$  is provided. In this differential characteristic, the input difference of the operation SubColumn in the  $2^{nd}$  round and the output difference of the operation SubColumn in the  $16^{th}$  round are as follows;



Table 6.6: Differential distributions of the S-box in REC-0

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	2	2	0	2	2	0	4	0	2	0	2
2	0	0	0	0	0	0	4	4	0	0	0	0	0	0	4	4
3	0	2	0	2	4	0	0	0	2	0	0	2	0	0	2	2
4	0	0	0	0	0	4	2	2	0	0	0	0	4	0	2	2
5	0	0	0	0	0	2	0	2	2	2	4	0	0	2	2	0
6	0	4	2	2	0	0	0	0	0	4	2	2	0	0	0	0
7	0	2	2	0	4	0	0	0	2	0	2	0	0	0	2	2
8	0	0	2	2	0	4	0	0	0	0	2	2	4	0	0	0
9	0	0	0	4	0	2	0	2	2	2	0	0	0	2	2	0
A	0	0	2	2	0	0	0	0	0	0	2	2	4	4	0	0
B	0	2	0	2	4	0	2	2	2	0	0	2	0	0	0	0
C	0	0	2	2	0	0	2	2	0	0	2	2	0	0	2	0
D	0	0	4	0	0	2	2	0	2	2	0	0	0	2	0	2
E	0	4	2	2	0	0	2	2	0	4	0	0	4	4	2	0
F	0	2	2	0	4	0	2	2	2	0	2	0	0	2	0	2

Table 6.7: n-round differential characteristics with fewer active S-boxes and the differences of the 2nd round and the 16th round subkeys are

Rounds	Number of active S-boxes	Probability
7*	7	$2^{-18}$
8*	10	$2^{-25}$
9*	12	$2^{-32}$
10	16	$2^{-41}$
11	19	$2^{-44}$
12	20	$2^{-51}$
13	23	$2^{-56}$
14	24	$2^{-59}$
15	26	$2^{-64}$

$$\Delta I_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\Delta O_{16} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

By fixing  $\Delta I_2$ ,  $\Delta O_{16}$ ,  $\Delta K_2$  and  $\Delta K_{16}$ , all 15-round differential characteristics with 26, 27, 28, 29 and 30 active S-boxes by using the method proposed in [50] are obtained, and the results are listed in Table 6.8. The total probability of all the differential characteristics in Table 6.8 is  $2^{-60.5}$ . This high differential probability with given  $\Delta I_2$ ,  $\Delta O_{16}$ ,  $\Delta K_2$  and  $\Delta K_{16}$  can be used to construct a distinguisher and recover partial secret keys. The attack progress is proposed in the next subsection.

Table 6.8: All 15-round differential characteristics with 26 – 30 active S-boxes

Number of active S-boxes	Number of diff. characteristics	Total probability
26	4	$2^{-62}$
27	30	$2^{-62}$
28	119	$2^{-62.82}$
29	324	$2^{-64.31}$
30	777	$2^{-65.97}$

### 6.3.1.2 A related-key differential attack on REC-0 with key length 80

In this section, we give a related-key differential attack on REC-0 with the key length 80 by using the differential characteristics in previous section and by extending 2 rounds backward and forward. To achieve this, we assume that the 19-round reduced RECTANGLE consists of 19 rounds of iterations and a final subkey XOR. We can number the rounds from 0 to 19. The  $0^{th}$  round subkey  $K_0$  is the master key and the final subkey is denoted by  $K_{19}$ . Now we can examine the attack in detail.

Firstly, we extend 2 rounds backward. The output difference  $\Delta O_1$  in the 1<sup>st</sup> round is obtained by taking inverses of AddRoundKey and ShiftRow operations with using the differences  $\Delta I_2$  and  $\Delta K_2$  as listed below;

$$\Delta O_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

We analyzed DDT of S-box of REC-0 and recognized that the output difference of the S-box is 1000 for the 10<sup>th</sup> column only if the input difference of the S-box is 1100, 0110, 1110, 1101, 0111 or 1111, and similar observations are found for the 3<sup>rd</sup> and 9<sup>th</sup> columns. Thus, the input difference  $\Delta I_1$  of the operation SubColumn in the 1<sup>st</sup> round should be in the following form;

$$\Delta I_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & ? & ? & 0 & 0 & 0 & 0 & 0 & 0 & ? & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & ? & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & ? & ? & 0 & 0 & 0 & 0 & 0 & ? & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & ? & ? & 0 & 0 & 0 & 0 & 0 & ? & 0 & 0 \end{pmatrix},$$

where the question mark denotes an undetermined value which can be 0 or 1. Totally,  $\Delta I_1$  has  $6 \times 7 \times 6 = 2^{7.98}$  cases. For a candidate  $\Delta I_1$  in the  $2^{7.98}$  cases, the probability of  $\Delta O_1$  in (5) is equal to  $\frac{1}{6} \times \frac{1}{7} \times \frac{1}{6} = 2^{-7.98}$ . By the key schedule, the difference of the 1<sup>st</sup> round subkey is as follows.

$$\Delta K_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

In a similar way, the output and input differences of the operation SubColumn in the 0<sup>th</sup> round are

$$\Delta O_0 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & ? & ? & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & ? & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & ? & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & ? & ? & ? & 0 & ? & ? & ? & ? & ? & 0 & 0 & ? & 0 \end{pmatrix},$$

$$\Delta I_0 = \begin{pmatrix} 0 & ? & ? & ? & 0 & ? & ? & ? & ? & ? & 0 & 0 & ? & 0 & 0 \\ 0 & ? & ? & ? & 0 & ? & ? & ? & ? & ? & 0 & 0 & ? & 0 & 0 \\ 0 & ? & ? & ? & 0 & ? & ? & ? & ? & ? & 0 & 0 & ? & 0 & 0 \\ 0 & ? & ? & ? & 0 & ? & ? & ? & ? & ? & 0 & 0 & ? & 0 & 0 \end{pmatrix}.$$

So, there are  $2^{7.98}$  cases for  $\Delta O_0$  and  $2^{36}$  cases for  $\Delta I_0$ . Therefore, for a random input difference  $\Delta I_0$ , the probability of the output difference  $\Delta O_0$  belonging to the  $2^{7.98}$  cases is  $2^{-28.02}$ . Besides, the differences of the  $0^{th}$  round subkey and the plaintexts are as follows:

$$\Delta K_0 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\Delta P = \begin{pmatrix} 0 & ? & ? & ? & 0 & ? & ? & ? & ? & ? & 0 & 0 & ? & 0 & 0 \\ 0 & ? & ? & ? & 0 & ? & ? & ? & ? & ? & 0 & 0 & ? & 0 & 0 \\ 0 & ? & ? & ? & 0 & ? & ? & ? & ? & ? & 0 & 0 & ? & 0 & 0 \\ 0 & ? & ? & ? & 0 & ? & ? & ? & ? & ? & 0 & 0 & ? & 0 & 0 \end{pmatrix},$$

In a similar way, we can extend 2 rounds forward. Since  $17^{th}$  round subkey is

$$\Delta K_{17} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

the input and output differences of the operation SubColumn in the  $17^{th}$  round are

$$\Delta I_{17} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\Delta O_{17} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & ? & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & ? \\ 0 & 0 & 0 & 0 & 0 & 0 & ? & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & ? \\ 0 & 0 & 0 & 0 & 0 & 0 & ? & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & ? & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & ? \end{pmatrix},$$

By the differential distributions of the S-box in Table 6.6, if the input difference of the S-box is  $\Delta I_S = 0010$ , the output difference of the S-box  $\Delta O_S = 1010, 0110, 1110, 0011, 0111, 1111$ . Therefore, the difference of the 18<sup>th</sup> round subkey is

$$\Delta K_{18} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

where  $* * 1* \in 1010, 0110, 1110, 0011, 0111, 1111$ . In this attack, it is assumed that the difference  $\Delta K_{18}$  is determined, since the attack is used six times for each value in the set  $1010, 0110, 1110, 0011, 0111, 1111$ . Then the differences of the operation SubColumn in the 18<sup>th</sup> round are

$$\Delta I_{17} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & ? & 0 & * & 0 & 0 & 0 & 0 & 0 & 0 & ? \\ 0 & 0 & 0 & 0 & 0 & ? & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & ? & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & ? & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & ? & 0 & 0 & 0 & 0 & 0 & ? & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\Delta O_{17} = \begin{pmatrix} 0 & 0 & ? & ? & ? & ? & ? & 0 & ? & ? & ? & 0 & 0 & 0 & ? & ? \\ 0 & 0 & ? & ? & ? & ? & ? & 0 & ? & ? & ? & 0 & 0 & 0 & ? & ? \\ 0 & 0 & ? & 1 & 1 & ? & ? & 0 & ? & ? & ? & 0 & 0 & 0 & ? & ? \\ 0 & 0 & ? & ? & ? & ? & ? & 0 & ? & ? & ? & 0 & 0 & 0 & ? & ? \end{pmatrix},$$

where the '\*' mark denotes a determined value in 0, 1 for a specific attack and the mark '?' determined an undetermined value in 0, 1. The number of  $\Delta O_{178}$  at most is  $2^{26.54}$ . Since the difference of the final subkey

$$\Delta K_{19} = \begin{matrix} 0 & 0 & 0 & 0 & 0 & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & * & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix},$$

is determined, the number of the difference of the ciphertext

$$\Delta K_{19} = \begin{matrix} 0 & * & ? & ? & ? & ? & ? & 0 & ? & ? & ? & 0 & 0 & 0 & ? & ? \\ 0 & ? & ? & ? & ? & ? & 0 & ? & ? & ? & 0 & 0 & 0 & ? & ? & 0 \\ 0 & 0 & ? & ? & 0 & 0 & ? & 1 & 1 & ? & ? & 0 & ? & ? & ? & 0 \\ 0 & ? & ? & 0 & 0 & ? & ? & ? & ? & ? & 0 & ? & ? & ? & 0 & 0 \end{matrix},$$

at most is  $2^{26.54}$ . Therefore, the probability of the ciphertext difference satisfying the form  $\Delta C$  is  $2^{-37.46}$

### Data Collection Phase

To find the expected number of the plaintext pairs corresponding to  $\Delta I_2$  and  $\Delta O_{16}$ , we should choose a  $2^x$ . For every structure,  $2^{36}$  plaintexts are captured by fixing the values in the  $0^{th}$ ,  $1^{th}$ ,  $2^{nd}$ ,  $4^{th}$ ,  $5^{th}$ ,  $11^{th}$  and  $15^{th}$  columns and traversing the values in the  $3^{rd}$ ,  $6^{th}$ ,  $7^{th}$ ,  $8^{th}$ ,  $9^{th}$ ,  $10^{th}$ ,  $12^{th}$ ,  $13^{th}$  and  $14^{th}$  columns. The  $2^{36}$  plaintexts can generate  $2^{72}$  pairs. A pair of plaintext in a structure can result in the expected input difference  $\Delta I_2$  with the probability of  $2^{-35.77}$ . Thus, the expected number of plaintext pairs corresponding to  $\Delta I_2$  and  $\Delta O_{16}$  is  $2^{x+72-36-60.5} = 2^{x-24.5}$

### Key Recovery Phase

For each structure, there are  $2^{72-37.46} = 2^{34.54}$  ordered pairs left according to the difference of the ciphertext  $\Delta C$ . Therefore, the expected number of remaining pairs is  $2^{x+34.54}$ .

The details of attack as follows;

- Step 1: Guess the value of a part of subkey bits of  $K_0$ .
  1. Guess  $K_0^{(3)}$  and compute the output difference of the 3<sup>rd</sup> S-box for each remaining plaintext pair; i.e

$$S(P^{(3)} \oplus K_0^{(3)}) \oplus S(P'^{(3)} \oplus S(K_0^{(3)} \oplus \Delta K_0^{(3)}))$$

If the difference do not have the form ?000, discard the pair. Then the number of expected remaining pairs is  $2^{x+31.54}$ .

2. Repeatedly guess  $K_0^{(6)}, K_0^{(7)}, K_0^{(8)}, K_0^{(9)}, K_0^{(10)}, K_0^{(12)}, K_0^{(13)}$  and  $K_0^{(14)}$ . There are  $2^{x+8.54}$  right pairs left.

- Step 2: Guess the value of a part of subkey bits of  $K_0$  by guessing some bits of  $K_0$  and  $K_1$ .

1. Since many bits of  $K_1$  are obtained from  $K_0$  directly by shifting and adding constant, we only need to guess some bits for a column in  $K_1$ . For the 3<sup>rd</sup> column of  $K_1$ , by the key schedule we have

$(K_1^{(0,3)}, K_1^{(1,3)}, K_1^{(2,3)}, K_1^{(3,3)}) = (K_0^{(0,16)}, K_0^{(1,14)}, K_0^{(2,12)}, K_0^{(3,10)})$  Therefore, we need to guess  $K_0^{(0,16)} = K_1^{(0,3)}$ . Then the number of expected remaining pairs is  $2^{x+4.54}$ .

2. Guess the bits  $K_0^{(1,1)}, K_0^{(2,19)}, K_0^{(3,17)}$ , and then check up whether

$$S(I_1^{(10)} \oplus K_1^{(10)}) \oplus S(I'^{(10)} \oplus K_1^{(10)} \oplus \Delta K_1^{(10)}) = 1000$$

since  $(K_1^{(0,10)}, K_1^{(1,10)}, K_1^{(2,10)}, K_1^{(3,10)}) = (K_0^{(0,3)}, K_0^{(1,1)}, K_0^{(2,19)}, K_0^{(3,17)})$

On average, there are  $2^{x+0.54}$  right pairs left.

3. Similarly, as Step 2(2), guess the bits  $K_0^{(0,2)}, K_1^{(1,9)}, K_0^{(2,18)}, K_0^{(3,16)}$ , then there are  $2^{x-3.46}$  right pairs left on average.

- Step 3: Guess the value of a part of subkey bits of  $K_{19}$ .

- For the 11<sup>th</sup> column of  $O_{18}$ , the secret bits  $K_{19}^{(0,11)}, K_{19}^{(1,12)}, K_{19}^{(2,7)}$  and  $K_{19}^{(3,8)}$  of  $K_{19}$  are involved. Guess the bits  $K_0^{(0,18)}, K_{19}^{(3,2)}, K_0^{(1,2)}, K_0^{(0,19)}$  and  $K_0^{(3,1)}$ , then combining with the guessed bits from Step 1 to Step 2, the involved secret bits  $K_{19}^{(0,11)}, K_{19}^{(1,12)}, K_{19}^{(2,7)}, K_{19}^{(3,8)}$  of  $K_{19}$  are determined. Then by using the method in Step 1.a, there are  $2^{x-7.46}$  right pairs left on average. Further, the bits  $K_{19}^{(0,12)}$

,  $K_{19}^{(1,13)}$ ,  $K_{19}^{(2,8)}$  and  $K_{19}^{(3,9)}$  are also determined, which are related to the 12<sup>th</sup> column of  $O_{18}$ . Then there are  $2^{x-11.46}$  right pairs left on average.

1. Guess  $K_0^{(1,16)}$ ,  $K_0^{(2,4)}$  and  $K_0^{(1,11)}$ , then combining with the guessed bits from Step 1 to Step 3(a), the secret bits  $K_{19}^{(0,1)}$ ,  $K_{19}^{(1,2)}$ ,  $K_{19}^{(2,13)}$  and  $K_{19}^{(3,14)}$  are determined, which are related to the 1st column of  $O_{18}$ . Then the number of remaining expected pairs is  $2^{x-14.46}$ .
  2. Similarly as Steps 3(a) and 3(b), we respectively guess the bits  $K_0^{(0,1)}$ ,  $K_0^{(3,19)}$  and  $K_0^{(0,4)}$  for the 6<sup>th</sup> column, the bits  $K_{19}^{(1,8)}$ ,  $K_0^{(1,18)}$  and  $K_0^{(2,2)}$  for the 7<sup>th</sup> column, the bit  $K_{19}^{(2,12)}$  for the 0<sup>th</sup> column, the bits  $K_{19}^{(0,9)}$ ,  $K_{19}^{(2,5)}$ ,  $K_0^{(0,17)}$ ,  $K_0^{(1,19)}$  and  $K_0^{(2,1)}$  for the 9th column, the bits  $K_{19}^{(2,6)}$  and  $K_{19}^{(3,7)}$  for the 10th column, the bits  $K_{19}^{(0,5)}$ ,  $K_{19}^{(1,6)}$  and  $K_{19}^{(3,2)}$  for the 5th column, and the bits  $K_{19}^{(0,13)}$ ,  $K_{19}^{(1,14)}$  and  $K_{19}^{(2,9)}$  for the 13th column. Then the number of remaining expected pairs is  $2^{x-36.46}$ .
- Step 4: The involved secret bits of  $K_{18}$  have guessed in Steps 1 – 3, and we do not need to guess any other secret bits. There are  $2^{x-44.46}$  right pairs left on average. Add one to the corresponding counter, if there is a right pair left.
  - Step 5: If the counter is larger than 1, keep the guess of the subkey bits as the candidates of the right subkeys. For each survived candidate, compute the seed key by doing an exhaustive search for other secret bits.

## Complexity Analysis

Since the expected number of the plaintext pairs corresponding to  $\Delta I_2$  and  $\Delta O_{16}$  is  $2^{x+72-36-60.5} = 2^{x-24.5}$ , we take  $x = 26$  such that the expected number can reach to 3. Therefore, the data complexity is  $2^{62}$ . To analyze the time complexity, we analyze the time complexity in each step. In the encryption phase, the time complexity is  $2^{63}$  19-round encryptions.

In Step 1.a, the time complexity is  $2 \times 2^{x+34.54} \times 2^4 \times \frac{1}{16} \times \frac{1}{19} \approx 2^{x+29.54}$  19-round encryptions. In Step 1.b, the time complexity is

$$2 \times (2^{x+39.54} + 2^{x+40.54} + 2^{x+41.54} + 2^{x+42.54} + 2^{x+43.54} + 2^{x+44.54})$$



$$+2^{x+45.54} + 2^{x+46.54}) \times \frac{1}{16} \times \frac{1}{19} \approx 2^{x+40.54}$$

In Step 2, the time complexity is  $2 \times (2^{x+45.54} + 2^{x+44.54} + 2^{x+44.54}) \times \frac{1}{16} \times \frac{1}{19} \approx 2^{x+39.54}$  19-round encryptions. In Step 3, the time complexity is

$$2 \times (2^{x+45.54} + 2^{x+41.54} + 2^{x+40.54} + 2^{x+40.54} + 2^{x+40.54} + 2^{x+37.54} \\ + 2^{x+39.54} + 2^{x+38.54} + 2^{x+38.54} + 2^{x+38.54}) \times \frac{1}{16} \times \frac{1}{19} \approx 2^{x+38.54}$$

In Step 4, the time complexity is about  $2^{x+28.54}$  19-round encryptions. Therefore, the total time complexity is  $2^{67.42}$ . The memory complexity is  $2^{72}$  key counters.

In conclusion, it is found out a huge number of 15-round related-key differential characteristics with the obtained input, output and round subkey differences. The total probability is  $2^{-60.5}$ . Based on these differential characteristics, a related-key differential attack on the 19-round reduced REC-0 by respectively extending 2 rounds backward and forward is found in [45], with a data complexity of  $2^{62}$ , a time complexity of  $2^{67.42}$  19-round encryptions and a memory complexity of  $2^{72}$ .

### 6.3.2 18-round Related-key Differential Attack on RECTANGLE

Revising the key schedule of REC-0 made RECTANGLE more secure against related-key attacks and the above 19-round related-key differential attack is not applicable to RECTANGLE. In the single key scenario, designers provided in [45] a 14-round difference propagation with the probability of  $2^{-62.83}$ . Designers claim that they can mount an attack on 18-round RECTANGLE using this 14-round characteristic without giving the exact details of this attack. This is the highest number of rounds the designers can break.



## CHAPTER 7

### CORRECTIONS AND IMPROVEMENTS ON DIFFERENTIAL ATTACKS ON RECTANGLE

In this chapter, we introduce the improvements and corrections to the two differential attacks on RECTANGLE and PRESENT.

#### 7.1 Improvement on 19-Round Related-Key Differential Attack on REC-0

As we mentioned in the previous chapter, differences of the  $2^{nd}$  round and the  $16^{th}$  round subkeys  $\Delta K_2$  and  $\Delta K_{16}$  are fixed in [45] to obtain differential characteristics. The input and output differences  $\Delta I_2$  and  $\Delta O_{18}$  are fixed as summarized in the Table 7.1. 19 rounds are attacked by adding two rounds to the top and the bottom of 1254 characteristics with a total probability of  $2^{60.5}$  is obtained MILP based methods.

Table 7.1: 19-round differential-linear attack of REC-0. differential factors are shown in bold.

Rounds	Differences in bits															
	$x_{15}$	$x_{14}$	$x_{13}$	$x_{12}$	$x_{11}$	$x_{10}$	$x_9$	$x_8$	$x_7$	$x_6$	$x_5$	$x_4$	$x_3$	$x_2$	$x_1$	$x_0$
$X_{0,I}$	0000	????	????	????	0000	????	????	????	????	????	0000	0000	????	0000	0000	0000
$X_{0,O}$	0000	<b>0?00</b>	??00	?000	0000	000?	001?	01?0	<b>0?00</b>	?000	0000	0000	000?	0000	0000	0000
$X_{1,I}$	0000	0000	0000	0000	0000	??1?	????	0000	0000	0000	0000	0000	??0?	0000	0000	0000
$X_{1,O}$	0000	0000	0000	0000	0000	0001	0010	0000	0000	0000	0000	0000	0101	0000	0000	0000
15-Round Differential $\Delta_1$																
$X_{17,I}$	0000	0000	0000	0000	0000	0000	0001	0000	0000	0000	0000	0000	0000	0000	0000	0100
$X_{17,O}$	0000	0000	0000	0000	0000	0000	????	0000	0000	0000	0000	0000	0000	0000	0000	?1??
$X_{18,I}$	0000	0000	?000	0100	0100	<b>00?0</b>	00*?	0000	000*	?000	0?00	0000	0000	0000	<b>00?0</b>	000?
$X_{18,O}$	0000	0000	????	?1??	?1??	????	????	0000	????	????	????	0000	0000	0000	????	????

Since REC-0 uses the inverse S-box of RECTANGLE, it has inverse differential  $\lambda = 4$  of

RECTANGLE for  $\mu = 2$  as in the figure 6.5 as explained in Theorem 4.2.7. Since these differential factors are two rounds away from the characteristic, Theorem 4.2.8 do not apply.

## 7.2 Experimental Differential Cryptanalysis

Although we know that the differential factors are two rounds away from characteristic we would like to analyze whether these differential factors affect the time complexity of the attack or not. While analyzing the effects of these differential factors, we implemented Java codes with several aims. These codes can be found in Appendix.

Our first aim is analyzing Rectangle algorithm and its differential cryptanalysis. To achieve this goal, we reduced differential attack to two rounds. While the key-schedule is used in RECTANGLE, we used two predetermined round keys which are  $rk_1$  and  $rk_2$  by ignoring key-schedule. After that, we take 100 O and O' pairs. We decrypted them 2 rounds with  $rk_1$  and  $rk_2$  and then encrypted them with  $rk_1$  to get I and I'. When we encrypted I and I' with  $rk_2$  we saw the results of the decryption were O and O'. In this way, we verified that we could successfully implemented RECTANGLE cipher.

After that, we assumed that we do not know the  $rk_2$ . Since there are 3 active S-boxes in this round, we can capture 12 bits of the key by differential cryptanalysis. Hence we considered  $2^{12}$  bits of the key to capture. We pick an array of  $2^{12}$  and one counter. Then we encrypted I and I' with every possible keys. When we observed O and O', we increased the counter. Finally, we saw that the counter value of the actual  $rk_2$  is 100. The other counter values of the wrong keys are smaller than 100. The steps of our experiment are as follows:

1. Take 100 many O and O' pairs.
2. Fix  $rk_1$  and  $rk_2$ .
3. Decrypt O and O' two round with  $rk_1$  and  $rk_2$  to get I and I'.

4. Encrypt  $I$  and  $I'$  with  $rk_1$  1 round to get  $I_2$   $I'_2$ .
5. Take a counter, and reset value as 0.
6. Encrypt  $I_2$   $I'_2$  with every possible key( only bits on the active S-boxes ) to get  $O_2$  and  $O'_2$ .
7. If  $O_2 \oplus O'_2$  equals to  $O \oplus O'$ , increase the counter value 1.
8. Compare the key bits which have higher hit(the value of counter is 100) with  $rk_2$ .

This is the last step of the differential cryptanalysis; which is the extracting key bits. In this part of analysis, the key bits on the active S-boxes can be captured by using differential characteristics. The remaining key bits are founded by exhaustive search as we introduced in Section 5.1.1.2. The total time complexity is summation of the two steps' complexities.

Our second aim is analyzing the effect of differential factors. Since we know that two differential factors are two rounds away from the characteristic, we did not expect to give any correction about the time complexity of the attack. Nevertheless, we want to analyze the two differential factors of RECTANGLE cipher.

First, we took 100  $O$  and  $O'$  pairs. Then we decrypted them with  $rk_1$  and  $rk_2$  to get  $I$  and  $I'$ . After that step, we encrypted  $I$  and  $I'$  with  $rk_1 \oplus \lambda$  instead of  $rk_1$ . If the differential factors were one round above, we would face with two keys which have same hits, i.e 100. Although, our differential factors are two round away from characteristic, we found again two keys that have the counter value of 100. The steps of our experiment is as follows:

1. Take 100 many  $O$  and  $O'$  pairs(  $\Delta I_2 = O \oplus O'$  ).
2. Fix  $rk_1$  and  $rk_2$ .
3. Decrypt  $O$  and  $O'$  two round with  $rk_1$  and  $rk_2$  to get  $I$  and  $I'$ .
4. Check  $I \oplus I' = \Delta I_0$ .
5. Take a counter, and reset value as 0.

6. Encrypte  $I$  and  $I'$  with  $(k_1 \oplus \lambda)$  and  $k_2$  then check whether the output difference is still  $\Delta I_2$ .

Up to this point, we used fixed keys both in two rounds to analyze RECTANGLE cipher easily. However, RECTANGLE cipher uses key-schedule to provide more security. Because of that reason, our experiment was not complete. We also implemented key schedule algorithm and also tried to capture key bits as we explained above. In this case, when we repeated the above experiment with two keys  $k_1$  and the  $k_2$  that is calculated by key-schedule, we recognized that there are four key pairs that have maximum counter value because of two differential factors. This was very important for us since we found two key pairs that have maximum counter value while the differential factors were two rounds away from the differential characteristic.

In addition to that, when we analyzing the key bits, we saw that differential factor  $\lambda = 4$  flips the value of the bit that corresponds to  $\mu = 2$ .

We summarized this finding as the following property.

**Property 7.1** *The differential factor  $\lambda = 4$  for  $\mu = 2$  flips the value of the bit that corresponds to  $\mu = 2$ . Namely, the second bits from the right of  $S(x)$  and  $S(y \oplus 4)$  are the same (similarly for  $S(y)$  and  $S(x \oplus 4)$ ).*

With this property, we can guess only half of the keys that correspond to the two S-boxes  $x_{14}$  and  $x_{17}$  in the first round. Therefore, if we start guessing keys from these two S-boxes, we reduce the time complexity of the first step of the attack by a factor of  $2^2$ . However, since the differential factors flips the values of the bits according to the above property, we need to also try to find the complements of the two key bits  $K_0^{(3,10)} = K_1^{(3,3)}$  and  $K_0^{(0,16)} = K_1^{(0,3)}$  in step 2 to avoid missing the correct key. With these corrections, steps of this time complexities of the modified attack changed as  $2^{x+38.29}$ ,  $2^{x+39.29}$ ,  $2^{x+38.55}$  and  $2^{x+28.54}$  19-round encryptions, respectively; you can find details of the steps of the new calculation as follows:

- Step 1: Guess the value of a part of subkey bits of  $K_0$ .
  1. Guess  $K_0^{(14)}$  and compute the output difference of the 14rd S-box for each

remaining plaintext pair; i.e

$$S(P^{(14)} \oplus K_0^{(14)}) \oplus S(P'^{(14)} \oplus S(K_0^{(14)} \oplus \Delta K_0^{(14)}))$$

This step has time complexity

$$2 \times 2^{x+34.54} \times 2^3 \times \frac{1}{16} \times \frac{1}{19} \approx 2^{x+31.54}$$

If the difference do not have the form ?000, discard the pair. Then the number of expected remaining pairs is  $2^{x+28.54}$ .

2. Guess  $K_0^{(7)}$  and compute the output difference of the 7<sup>th</sup> S-box for each remaining plaintext pair; i.e

$$S(P^{(7)} \oplus K_0^{(7)}) \oplus S(P'^{(7)} \oplus K_0^{(7)} \oplus \Delta K_0^{(7)})$$

This step has time complexity

$$2 \times 2^{x+31.54} \times 2^6 \times \frac{1}{16} \times \frac{1}{19} \approx 2^{x+28.54}$$

If the difference do not have the form ?000, discard the pair. Then the number of expected remaining pairs is  $2^{x+28.54}$ .

3. Repeatedly guess  $K_0^{(3)}, K_0^{(6)}, K_0^{(8)}, K_0^{(9)}, K_0^{(10)}, K_0^{(12)}, K_0^{(13)}$ . There are  $2^x + 7.54$  right pairs left. This step has time complexity

$$2 \times (2^{x+36.54} + 2^{x+37.54} + 2^{x+38.54} + 2^{x+39.54} + 2^{x+40.54} + 2^{x+41.54} + 2^{x+42.54} + 2^{x+43.54}) \times \frac{1}{16} \times \frac{1}{19} \approx 2^{x+38.54}$$

- Step 2: Guess the value of a part of subkey bits of  $K_0$  by guessing some bits of  $K_0$  and  $K_1$ .

1. Since many bits of  $K_1$  are obtained from  $K_0$  directly by shifting and adding constant, we only need to guess some bits for a column in  $K_1$ . For the 3rd column of  $K_1$ , by the key schedule we have

$(K_1^{(0,3)}, K_1^{(1,3)}, K_1^{(2,3)}, K_1^{(3,3)}) = (K_0^{(0,16)}, K_0^{(1,14)}, K_0^{(2,12)}, K_0^{(3,10)})$  Therefore, we need to guess  $K_0^{(0,16)} = K_1^{(0,3)}$  and we also need  $K_0^{(3,10)} = K_1^{(3,3)}$  because  $K_1^{(3,3)}$  was flipped when we apply Substitution operation to  $K_1^{(2,7)}$ . Then the number of expected remaining pairs is  $2^{x+5.54}$ .

2. Guess the bits  $K_0^{(1,1)}$ ,  $K_0^{(2,19)}$ ,  $K_0^{(3,17)}$ , and then check up whether

$$S(I_1^{(10)} \oplus K_1^{(10)}) \oplus S(I_1^{(10)} \oplus K_1^{(10)} \oplus \Delta K_1^{(10)}) = 1000$$

since  $(K_1^{(0,10)}, K_1^{(1,10)}, K_1^{(2,10)}, K_1^{(3,10)}) = (K_0^{(0,3)}, K_0^{(1,1)}, K_0^{(2,19)}, K_0^{(3,17)})$

On average, there are  $2^{x+1.54}$  right pairs left.

3. Similarly, as Step 2(2), guess the bits  $K_0^{(0,2)}$ ,  $K_1^{(1,9)}$ ,  $K_0^{(2,18)}$ ,  $K_0^{(3,16)}$ , then there are  $2^{x-2.46}$  right pairs left on average.

In step 2, time complexity is

$$2 \times (2^{x+44.54} + 2^{x+43.54} + 2^{x+44.54}) + \frac{1}{16} \times \frac{1}{19} \times 2^{x+38.54}$$

As in [45] if we choose  $x = 26$ , we get a time complexity of  $2^{66.35}$  19-round encryptions compared to  $2^{67.42}$  of the original attack. Thus, the time complexity of the attack reduces with factor of  $2^{1.07}$  by using differential factors.

We further give a correction to this attack due to the undisturbed bits. For  $\Delta I_2$  the S-box only activates third bit, it was assumed that we need to capture the values of one S-box in the first round. However, we cannot verify the characteristic without knowing the all four bits of the S-box output in the 1<sup>st</sup> round. We provided the parts that need to be obtained in bold in Table 7.2. Thus, the attacker also needs to guess the 1<sup>st</sup> round subkey corresponding to  $S_2$ . But the attackers advantage increases by 3 instead of 4 bits due to the following property.

**Property 7.2** *Inverse of RECTANGLE's S-box S has the property  $tb(S(x)) = tb(S(x \oplus E))$  where  $tb$  is the third bit.*

Since  $K_0^{(0,2)}$  was guessed in Step 2.3, the time complexity in Step 2 increase like in the following way;

$$2 \cdot (2^{x+45.54} + 2^{x+45.54} + 2^{x+44.54}) \cdot \frac{1}{16} \cdot \frac{1}{19} \approx 2^{x+39.54}$$

In addition, the time complexity of Step 1 is changed in the following way;

$$2 \cdot (2^{x+36.54} + 2^{x+37.54} + 2^{x+38.54} + 2^{x+39.54} + 2^{x+40.54} + 2^{x+41.54} + 2^{x+42.54} + 2^{x+44.54}) \cdot \frac{1}{16} \cdot \frac{1}{19} \approx 2^{x+39.54}$$



Thus, the whole 80-bit key can be obtained after a  $2^{67.35}$  19-round RECTANGLE encryptions instead of  $2^{67.42}$  encryptions. While these two complexities are very close to each other and it is seem to be that there is no improvement on the attack, we have corrected this attack by the help of the undisturbed bit. If the attackers tested this attack practically, they show the attack could not be done due to undisturbed bit.

Table 7.2: 19-round differential-linear attack of REC-0. Bits that should be captured because of undisturbed bit in the 1<sup>st</sup> are shown in bold.

Rounds	Differences in bits															
	$x_{15}$	$x_{14}$	$x_{13}$	$x_{12}$	$x_{11}$	$x_{10}$	$x_9$	$x_8$	$x_7$	$x_6$	$x_5$	$x_4$	$x_3$	$x_2$	$x_1$	$x_0$
$X_{0,I}$	0000	????	????	????	0000	????	????	????	????	????	0000	0000	????	<b>0000</b>	0000	0000
$X_{0,O}$	0000	0?00	??00	?000	0000	000?	001?	01?0	0?00	?000	0000	0000	000?	<b>0000</b>	0000	0000
$X_{1,I}$	0000	0000	0000	0000	0000	??1?	????	0000	0000	0000	0000	0000	??0?	0000	0000	0000
$X_{1,O}$	0000	0000	0000	0000	0000	0001	0010	0000	0000	0000	0000	0000	0101	0000	0000	0000
15-Round Differential $\Delta_1$																
$X_{17,I}$	0000	0000	0000	0000	0000	0000	0001	0000	0000	0000	0000	0000	0000	0000	0000	0100
$X_{17,O}$	0000	0000	0000	0000	0000	0000	????	0000	0000	0000	0000	0000	0000	0000	0000	?1??
$X_{18,I}$	0000	0000	?000	0100	0100	00?0	00*?	0000	000*	?000	0?00	0000	0000	0000	00?0	000?
$X_{18,O}$	0000	0000	????	?1??	?1??	????	????	0000	????	????	????	0000	0000	0000	????	????

### 7.3 Improvement on 18-Round Differential Attack on RECTANGLE

After 19-round attack designers of RECTANGLE proposed another attack to break 18-round of RECTANGLE by using 14 round characteristics [69]. The 14-round characteristics contain also two differential factors since the inverse S-box of REC-0 is used. The differential factors are as shown in the Table 7.3.

Table 7.3: The input difference and the output difference of the 14-round difference propagation

Input Difference of Round 0	Output Difference of Round 13
0000000000000000	0000000000000000
0010000100000000	0000000000000010
0000000100000000	0001000000000000
0000000000000000	0000000000000000

Since the details of the attack is not given, we can not give exact numbers about improvements. However, we can say that the 14-round characteristics contain two differential factors as shown in Table 7.3 and attacks on RECTANGLE using this or sim-

ilar characteristics should consider the effects of differential factors.

The time complexity of the 18-round attack is given as  $2^{78.67}$  18-round encryptions for an 80-bit seed key and  $2^{126.66}$  18-round encryptions for a 128-bit seed key. These complexities are formed by calculations of two processes; extracting key bits by using differential and trying all possible key bits by using exhaustive search for the remaining bits. One of them can be marginally small according to other; in this case this part can be negligible. On the other hand, these two calculations are very close to each other.

Since we do not know the details of that attack we do not deduce the exact effect of differential factors to attack. However we can say that if the first part of attack needs more complexity, the attack can be performed with the complexity between  $2^{76.67}$  and  $2^{80.67}$ . On the other hand, if the second part of the attack needs more complexity then the attack can be performed with the complexity between  $2^{124.66}$  and  $2^{128.66}$ .

## CHAPTER 8

### CONCLUSION

With the development of IoT technology, the need for lightweight block ciphers has increased. Hence, a lot of new block ciphers were introduced and still continue to be introduced. Against these ciphers, several attacks performed theoretically. Because those attacks are theoretical, they may contain incorrect results when they are investigated in practice. Dr. Tezcan recognized differential factors in 2014 in [58] while examining a theoretical attack on PRESENT cipher. Until the differential factors and undisturbed bits were introduced, cryptanalysis experts believed that they could capture the whole key bits corresponding to the active S-boxes in a differential attack. However, it came up that it could not be possible if these active S-boxes have differential factors. Thus, the correct key can be obtained by eliminating all or most of the wrong keys; that also means reducing key space and time complexity.

In this thesis, we investigated the theory of differential factors, undisturbed bits and their effects on 19 round differential attack on REC-0 cipher, 18-round differential attack on RECTANGLE and 16-round differential attack on PRESENT. As a result of these investigations, the time complexity of the 19-round related-key differential attack of [45] on the initial version of REC-0 decreased by a factor of  $2^{1.07}$  with the help of differential factors. We also showed that the attackers should be captured 3 more bits due to undisturbed bits. In addition to that we revealed some mistakes on 18-round differential attack on RECTANGLE but we could not give exact time complexities of the attack since we do not know the details of the attack. At the end, we showed that 16-round differential attack on PRESENT captures 32 bits instead of 40 bits. Therefore, we said that the remaining 48 bits requires  $2^{48}$  16-round PRESENT encryptions. We verified

also all these improvements experimentally. We completed this study in a search group. Not only REVTANGLE and PRESENT examined, but PRIDE was examined from the other person in the search group also. We published all findings in [57] and [56] briefly. Consequently, verifying theoretical attacks experimentally is so important to check their effectiveness in practice. Although it is not feasible because of the time, data and memory complexities, we are able to verify the theoretical results by using reduced versions of them. We believe that cryptanalysis would benefit from the practice of verifying theoretical results by experimenting on the reduced versions.



## REFERENCES

- [1] M. A. Abdelraheem, G. Leander, and E. Zenner. Differential cryptanalysis of round-reduced printcipher: Computing roots of permutations. In A. Joux, editor, *Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers*, volume 6733 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2011.
- [2] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita. Camellia: A 128-bit block cipher suitable for multiple platforms - design and analysis. In D. R. Stinson and S. E. Tavares, editors, *Selected Areas in Cryptography, 7th Annual International Workshop, SAC 2000, Waterloo, Ontario, Canada, August 14-15, 2000, Proceedings*, volume 2012 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2000.
- [3] J. Aumasson, M. Naya-Plasencia, and M. O. Saarinen. Practical attack on 8 rounds of the lightweight block cipher KLEIN. In D. J. Bernstein and S. Chatterjee, editors, *Progress in Cryptology - INDOCRYPT 2011 - 12th International Conference on Cryptology in India, Chennai, India, December 11-14, 2011. Proceedings*, volume 7107 of *Lecture Notes in Computer Science*, pages 134–145. Springer, 2011.
- [4] E. Biham. New types of cryptanalytic attacks using related keys. *J. Cryptology*, 7(4):229–246, 1994.
- [5] E. Biham, R. J. Anderson, and L. R. Knudsen. Serpent: A new block cipher proposal. In S. Vaudenay, editor, *Fast Software Encryption, 5th International Workshop, FSE '98, Paris, France, March 23-25, 1998, Proceedings*, volume 1372 of *Lecture Notes in Computer Science*, pages 222–238. Springer, 1998.
- [6] E. Biham, A. Biryukov, and A. Shamir. Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In J. Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 12–23. Springer, 1999.
- [7] E. Biham, O. Dunkelman, and N. Keller. Linear cryptanalysis of reduced round serpent. In M. Matsui, editor, *Fast Software Encryption, 8th International Workshop, FSE 2001 Yokohama, Japan, April 2-4, 2001, Revised Papers*, volume 2355 of *Lecture Notes in Computer Science*, pages 16–27. Springer, 2001.

- [8] A. Biryukov and D. Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In M. Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2009.
- [9] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Viskkelsoe. PRESENT: an ultra-lightweight block cipher. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.
- [10] C. D. Cannière, O. Dunkelman, and M. Knezevic. KATAN and KTANTAN - A family of small and efficient hardware-oriented block ciphers. In C. Clavier and K. Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*, pages 272–288. Springer, 2009.
- [11] Z. Chen and X. Wang. Impossible differential cryptanalysis of midori. *IACR Cryptology ePrint Archive*, 2016:535, 2016.
- [12] H. Cheng, H. M. Heys, and C. Wang. PUFFIN: A novel compact block cipher targeted to embedded digital systems. In L. Fanucci, editor, *11th Euromicro Conference on Digital System Design: Architectures, Methods and Tools, DSD 2008, Parma, Italy, September 3-5, 2008*, pages 383–390. IEEE Computer Society, 2008.
- [13] M. Çoban, F. Karakoç, and Ö. Boztas. Biclique cryptanalysis of TWINE. *IACR Cryptology ePrint Archive*, 2012:422, 2012.
- [14] S. Contini, R. L. Rivest, M. J. B. Robshaw, and Y. L. Yin. Improved analysis of some simplified variants of RC6. In L. R. Knudsen, editor, *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999, Proceedings*, volume 1636 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1999.
- [15] N. Courtois and J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. *IACR Cryptology ePrint Archive*, 2002:44, 2002.
- [16] P. Crowley. Truncated differential cryptanalysis of five rounds of salsa20. *IACR Cryptology ePrint Archive*, 2005:375, 2005.
- [17] J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

- [18] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, 22(6):644–654, 1976.
- [19] S. R. Fluhrer and S. Lucks. Analysis of the  $e_0$  encryption system. In S. Vaudenay and A. M. Youssef, editors, *Selected Areas in Cryptography, 8th Annual International Workshop, SAC 2001 Toronto, Ontario, Canada, August 16-17, 2001, Revised Papers*, volume 2259 of *Lecture Notes in Computer Science*, pages 38–48. Springer, 2001.
- [20] T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1984.
- [21] T. Gendrullis, M. Novotný, and A. Rupp. A real-world attack breaking A5/1 within hours. *IACR Cryptology ePrint Archive*, 2008:147, 2008.
- [22] Z. Gong, S. Nikova, and Y. W. Law. KLEIN: A new family of lightweight block ciphers. In A. Juels and C. Paar, editors, *RFID. Security and Privacy - 7th International Workshop, RFIDSec 2011, Amherst, USA, June 26-28, 2011, Revised Selected Papers*, volume 7055 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2011.
- [23] J. Guo, T. Peyrin, A. Poschmann, and M. J. B. Robshaw. The LED block cipher. In B. Preneel and T. Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*, pages 326–341. Springer, 2011.
- [24] M. E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Trans. Information Theory*, 26(4):401–406, 1980.
- [25] T. Isobe and K. Shibutani. Security analysis of the lightweight block ciphers xtea, LED and piccolo. In W. Susilo, Y. Mu, and J. Seberry, editors, *Information Security and Privacy - 17th Australasian Conference, ACISP 2012, Wollongong, NSW, Australia, July 9-11, 2012. Proceedings*, volume 7372 of *Lecture Notes in Computer Science*, pages 71–86. Springer, 2012.
- [26] K. Jia and N. Wang. Impossible differential cryptanalysis of 14-round camellia-192. In J. K. Liu and R. Steinfeld, editors, *Information Security and Privacy - 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part II*, volume 9723 of *Lecture Notes in Computer Science*, pages 363–378. Springer, 2016.
- [27] J. Kim, S. Hong, S. Lee, J. H. Song, and H. Yang. Truncated differential attacks on 8-round CRYPTON. In J. I. Lim and D. H. Lee, editors, *Information*

*Security and Cryptology - ICISC 2003, 6th International Conference, Seoul, Korea, November 27-28, 2003, Revised Papers*, volume 2971 of *Lecture Notes in Computer Science*, pages 446–456. Springer, 2003.

- [28] S. Knellwolf, W. Meier, and M. Naya-Plasencia. Conditional differential cryptanalysis of nlfsr-based cryptosystems. In M. Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 130–145. Springer, 2010.
- [29] S. Knellwolf, W. Meier, and M. Naya-Plasencia. Conditional differential cryptanalysis of trivium and KATAN. In A. Miri and S. Vaudenay, editors, *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, volume 7118 of *Lecture Notes in Computer Science*, pages 200–212. Springer, 2011.
- [30] L. R. Knudsen. Truncated and higher order differentials. In Preneel [40], pages 196–211.
- [31] L. R. Knudsen, G. Leander, A. Poschmann, and M. J. B. Robshaw. Printcipher: A block cipher for ic-printing. In S. Mangard and F. Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 16–32. Springer, 2010.
- [32] Y. Ko, D. Hong, S. Hong, S. Lee, and J. Lim. Linear cryptanalysis of SPECTR-H64 with higher order differential property. In V. Gorodetsky, L. J. Popyack, and V. A. Skormin, editors, *Computer Network Security, Second International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security, MMM-ACNS 2003, St. Petersburg, Russia, September 21-23, 2003, Proceedings*, volume 2776 of *Lecture Notes in Computer Science*, pages 298–307. Springer, 2003.
- [33] G. Leander. On linear hulls, statistical saturation attacks, PRESENT and a cryptanalysis of PUFFIN. In Paterson [39], pages 303–322.
- [34] G. Leander, M. A. Abdelraheem, H. AlKhzaimi, and E. Zenner. A cryptanalysis of printcipher: The invariant subspace attack. In P. Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 206–221. Springer, 2011.
- [35] G. Leander, C. Paar, A. Poschmann, and K. Schramm. New lightweight DES variants. In A. Biryukov, editor, *Fast Software Encryption, 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised*



- Selected Papers*, volume 4593 of *Lecture Notes in Computer Science*, pages 196–210. Springer, 2007.
- [36] D. Meeks. Hieroglyphic dictionary, inventory of hieroglyphs and unicode. *Document Numérique*, 16(3):31–44, 2013.
- [37] F. Mendel, V. Rijmen, D. Toz, and K. Varici. Differential analysis of the LED block cipher. *IACR Cryptology ePrint Archive*, 2012:544, 2012.
- [38] D. Moon, K. Hwang, W. Lee, S. Lee, and J. Lim. Impossible differential cryptanalysis of reduced round XTEA and TEA. In J. Daemen and V. Rijmen, editors, *Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers*, volume 2365 of *Lecture Notes in Computer Science*, pages 49–60. Springer, 2002.
- [39] K. G. Paterson, editor. *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*. Springer, 2011.
- [40] B. Preneel, editor. *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*, volume 1008 of *Lecture Notes in Computer Science*. Springer, 1995.
- [41] B. Reichardt and D. A. Wagner. Markov truncated differential cryptanalysis of skipjack. In K. Nyberg and H. M. Heys, editors, *Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002. Revised Papers*, volume 2595 of *Lecture Notes in Computer Science*, pages 110–128. Springer, 2002.
- [42] R. L. Rivest. The RC5 encryption algorithm. In Preneel [40], pages 86–96.
- [43] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.
- [44] J. Seberry, X. Zhang, and Y. Zheng. Pitfalls in designing substitution boxes (extended abstract). In Y. Desmedt, editor, *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, volume 839 of *Lecture Notes in Computer Science*, pages 383–396. Springer, 1994.
- [45] J. Shan, L. Hu, L. Song, S. Sun, and X. Ma. Related-key differential attack on round reduced RECTANGLE-80. *IACR Cryptology ePrint Archive*, 2014:986, 2014.
- [46] C. E. Shannon. Communication theory - exposition of fundamentals. *Trans. of the IRE Professional Group on Information Theory (TIT)*, 1:44–47, 1953.

- [47] H. Soleimany. Zero-correlation linear cryptanalysis of reduced-round lblock. *IACR Cryptology ePrint Archive*, 2012:570, 2012.
- [48] B. Sun, R. Li, M. Wang, P. Li, and C. Li. Impossible differential cryptanalysis of CLEFIA. *IACR Cryptology ePrint Archive*, 2008:151, 2008.
- [49] S. Sun, L. Hu, K. Qiao, X. Ma, J. Shan, and L. Song. Improvement on the method for automatic differential analysis and its application to two lightweight block ciphers DESL and lblock-s. In K. Tanaka and Y. Suga, editors, *Advances in Information and Computer Security - 10th International Workshop on Security, IWSEC 2015, Nara, Japan, August 26-28, 2015, Proceedings*, volume 9241 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 2015.
- [50] S. Sun, L. Hu, M. Wang, P. Wang, K. Qiao, X. Ma, D. Shi, and L. Song. Automatic enumeration of (related-key) differential and linear characteristics with predefined properties and its applications. *IACR Cryptology ePrint Archive*, 2014:747, 2014.
- [51] T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi.  $\text{\textnormal{\textsc{TWINE}}}$  : A lightweight block cipher for multiple platforms. In L. R. Knudsen and H. Wu, editors, *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, volume 7707 of *Lecture Notes in Computer Science*, pages 339–354. Springer, 2012.
- [52] S. E. Tavares and H. Meijer, editors. *Selected Areas in Cryptography '98, SAC'98, Kingston, Ontario, Canada, August 17-18, 1998, Proceedings*, volume 1556 of *Lecture Notes in Computer Science*. Springer, 1999.
- [53] C. Tezcan. Improbable differential cryptanalysis. In A. Elçi, M. S. Gaur, M. A. Orgun, and O. B. Makarevich, editors, *The 6th International Conference on Security of Information and Networks, SIN '13, Aksaray, Turkey, November 26-28, 2013*, page 457. ACM, 2013.
- [54] C. Tezcan. Improbable differential attacks on present using undisturbed bits. *J. Computational Applied Mathematics*, 259:503–511, 2014.
- [55] C. Tezcan. Differential factors revisited: Corrected attacks on PRESENT and SERPENT. In T. Güneysu, G. Leander, and A. Moradi, editors, *Lightweight Cryptography for Security and Privacy - 4th International Workshop, LightSec 2015, Bochum, Germany, September 10-11, 2015, Revised Selected Papers*, volume 9542 of *Lecture Notes in Computer Science*, pages 21–33. Springer, 2015.
- [56] C. Tezcan, G. O. Okan, A. Şenol, E. Doğan, F. Yücebaş, and N. Baykal. On differential factors. ISCTURKEY 2016, Ankara, Turkey, 103-110.

- [57] C. Tezcan, G. O. Okan, A. Senol, E. Dogan, F. Yücebas, and N. Baykal. Differential attacks on lightweight block ciphers present, pride, and RECTANGLE revisited. In A. Bogdanov, editor, *Lightweight Cryptography for Security and Privacy - 5th International Workshop, LightSec 2016, Aksaray, Turkey, September 21-22, 2016, Revised Selected Papers*, volume 10098 of *Lecture Notes in Computer Science*, pages 18–32. Springer, 2016.
- [58] C. Tezcan and F. Özbudak. Differential factors: Improved attacks on SERPENT. In T. Eisenbarth and E. Öztürk, editors, *Lightweight Cryptography for Security and Privacy - Third International Workshop, LightSec 2014, Istanbul, Turkey, September 1-2, 2014, Revised Selected Papers*, volume 8898 of *Lecture Notes in Computer Science*, pages 69–84. Springer, 2014.
- [59] C. Tezcan and A. A. Selçuk. Improved improbable differential attacks on ISO standard CLEFIA: expansion technique revisited. *Inf. Process. Lett.*, 116(2):136–143, 2016.
- [60] M. Tolba, A. Abdelkhalek, and A. M. Youssef. Impossible differential cryptanalysis of reduced-round SKINNY. In M. Joye and A. Nitaj, editors, *Progress in Cryptology - AFRICACRYPT 2017 - 9th International Conference on Cryptology in Africa, Dakar, Senegal, May 24-26, 2017, Proceedings*, volume 10239 of *Lecture Notes in Computer Science*, pages 117–134, 2017.
- [61] M. S. Turan and E. Uyan. Near-collisions for the reduced round versions of some second round SHA-3 compression functions using hill climbing. In G. Gong and K. C. Gupta, editors, *Progress in Cryptology - INDOCRYPT 2010 - 11th International Conference on Cryptology in India, Hyderabad, India, December 12-15, 2010. Proceedings*, volume 6498 of *Lecture Notes in Computer Science*, pages 131–143. Springer, 2010.
- [62] M. Wang. Differential cryptanalysis of reduced-round PRESENT. In S. Vaudey, editor, *Progress in Cryptology - AFRICACRYPT 2008, First International Conference on Cryptology in Africa, Casablanca, Morocco, June 11-14, 2008. Proceedings*, volume 5023 of *Lecture Notes in Computer Science*, pages 40–49. Springer, 2008.
- [63] Y. Wang, W. Wu, X. Yu, and L. Zhang. Security on lblock against biclique cryptanalysis. In D. H. Lee and M. Yung, editors, *Information Security Applications - 13th International Workshop, WISA 2012, Jeju Island, Korea, August 16-18, 2012, Revised Selected Papers*, volume 7690 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2012.
- [64] L. Wei, C. Rechberger, J. Guo, H. Wu, H. Wang, and S. Ling. Improved meet-in-the-middle cryptanalysis of KTANTAN. *IACR Cryptology ePrint Archive*, 2011:201, 2011.

- [65] D. J. Wheeler and R. M. Needham. Tea, a tiny encryption algorithm. In Preneel [40], pages 363–366.
- [66] W. Wu and L. Zhang. Lblock: A lightweight block cipher. *IACR Cryptology ePrint Archive*, 2011:345, 2011.
- [67] H. Yap, K. Khoo, A. Poschmann, and M. Henricksen. EPCBC - A block cipher suitable for electronic product code encryption. In D. Lin, G. Tsudik, and X. Wang, editors, *Cryptology and Network Security - 10th International Conference, CANS 2011, Sanya, China, December 10-12, 2011. Proceedings*, volume 7092 of *Lecture Notes in Computer Science*, pages 76–97. Springer, 2011.
- [68] W. Zhang, Z. Bao, D. Lin, V. Rijmen, B. Yang, and I. Verbauwhede. RECTANGLE: A bit-slice ultra-lightweight block cipher suitable for multiple platforms. *IACR Cryptology ePrint Archive*, 2014:84, 2014.
- [69] W. Zhang, Z. Bao, D. Lin, V. Rijmen, B. Yang, and I. Verbauwhede. RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. *SCIENCE CHINA Information Sciences*, 58(12):1–15, 2015.
- [70] G. Zhao, B. Sun, C. Li, and J. Su. Truncated differential cryptanalysis of PRINCE. *Security and Communication Networks*, 8(16):2875–2887, 2015.

## APPENDIX A

### APPENDIX I

```
package bruteforce;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.Arrays;

public class BruteForce
{
    //this is a random key1
    static boolean[] booleanKey={true,false,false,true,true,true,
    false, false,false,false,false,false,false,true,
    true,false,true,true,true,true,false,true,true,false,false,
    false,false,true,false,true,true,true,true,false,false,
    false,true,true,true,false,false,false,false,false,true,
    false,true,false,false,false,false,true,true,true,false,false,
    false,false,false,false};

    //this is a random key2
    static boolean[] booleanKey2 = {false,true,false,false,false,
    true,false,true,false,false,true,true,true,false,false,
    false,false,true,false,true,true,false,false,true,false,false,
```

```
true,true,true,true,false,true,false,false,false,false,false,
false,true,false,false,true,true,true,true,true,false,true,
true,true,false,false,false,false,false,false,false,false,
false,true,false,false,true};
```

```
//this is a random key3
```

```
static boolean[] booleanKey3 ={true,false,false,true,true,true,
false, false,false,false,false,false,false,false,true,
true,false,true,true,true,true,false,true,true,false,false,
false,false,true,true,true,true,true,true,true,false,false,
false,true,true,true,false,false,false,false,false,true,
false,true,false,false,false,false,true,true,true,false,false,
false,false,false,false};
```

```
//this is DeltaI2 from characteristic
```

```
static boolean[] InputDifference2={false,false,false,true,
false,false,false,false,false,false,true,false,false,false,
false,false,false,false,false,false,false,false,false,false,
false,false,true,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false};
```

```
//this is DeltaI1 from characteristic
```

```
static boolean[] booleanKeyDifference1 = {false,false,false,
false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,true,false,
false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,
false};
```

```

//this is for key schedule, DeltaK2
static boolean[] booleanKeyDifference2 = {false,false,false,
false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,
false,false,false,true,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,
false};

public static void main(String[] args) throws IOException {

boolean[][] arrayCiphertext = readTextFile("ciphertexts.txt");
int[] counters = new int[4096];

//starting of key schedule
boolean[] diffKey = addKeyOperation(booleanKeyDifference1,booleanKey);
boolean[] diffKey3 = addKeyOperation(booleanKeyDifference1,booleanKey3);
boolean[] diffKey2 = addKeyOperation(booleanKeyDifference2,
booleanKey2);
boolean[] manipulatedKey2 = booleanKey2.clone();
boolean[] manipulatedDiffKey2 = diffKey2.clone();
//end of key schedule

for (int i = 0; i < 100; i++)
{
    boolean[] encryptedMatrix = arrayCiphertext[i];
    boolean[] encryptedDiffMatrix = addKeyOperation(
    encryptedMatrix,
    InputDifference2);

    boolean[] decryptedMatrix =
    Decryption(booleanKey, booleanKey2,

```

```

encryptedMatrix);
boolean[] decryptedDiffMatrix =
Decryption(diffKey, diffKey2,
encryptedDiffMatrix);

boolean[] encryptedSubMatrix = Encryption(booleanKey3,
decryptedMatrix);
boolean[] encryptedSubDiffMatrix = Encryption(diffKey3,
decryptedDiffMatrix);

for (int k = 0; k < 4096; k++)
{
    ManipulateKey_12Bit(manipulatedKey2, k);
    ManipulateKey_12Bit(manipulatedDiffKey2, k);

    boolean[] testOutputMatrix = Encryption(manipulatedKey2,
encryptedSubMatrix);
    boolean[] testOutputDiffMatrix =
Encryption(manipulatedDiffKey2,
encryptedSubDiffMatrix);

    boolean[] testOutputDeltaMatrix =
addKeyOperation(testOutputMatrix,
testOutputDiffMatrix);

    if (CheckMatrixEquality(testOutputDeltaMatrix, InputDifference2))
    {
        counters[k] += 1;
    }
}
}

```



```

for (int i = 0; i < 4096; i++)
{
    System.out.println("----- Key [" + i + "]
    -----");
    System.out.println("Counter Result : " + counters[i]);

    ManipulateKey_12Bit(manipulatedKey2, i);
    PrintMatrix2("Key Data", manipulatedKey2);

    System.out.print("\n\n\n");
}

PrintMatrix2("Key 2", booleanKey2);
PrintMatrix2("Key 1", booleanKey);
PrintMatrix2("Key 3", booleanKey3);
}

public static void ManipulateKey_12Bit(boolean[] manipulatedKey,
int manipulateIndex)
{
    boolean[] bitValues = new boolean[12];

    for (int b = 0; b < 12; b++)
    {
        bitValues[b] = (manipulateIndex & (1 << b)) != 0;
    }

    manipulatedKey[3] = bitValues[0];
    manipulatedKey[19] = bitValues[1];
    manipulatedKey[35] = bitValues[2];
    manipulatedKey[51] = bitValues[3];
}

```

```

    manipulatedKey[10] = bitValues[4];
    manipulatedKey[26] = bitValues[5];
    manipulatedKey[42] = bitValues[6];
    manipulatedKey[58] = bitValues[7];

    manipulatedKey[9] = bitValues[8];
    manipulatedKey[25] = bitValues[9];
    manipulatedKey[41] = bitValues[10];
    manipulatedKey[57] = bitValues[11];
}

// this is for checking two matrix equality
public static boolean CheckMatrixEquality(boolean[] matrix1,
boolean[] matrix2)
{

    int matrixLength = matrix1.length;

    if (matrixLength != matrix2.length)
    {
        System.out.println("Different matrix sizes
at equality check!");
        return false;
    }

    for (int i = 0; i < matrixLength; i++)
    {
        if (matrix1[i] != matrix2[i])
            return false;
    }

    return true;
}

```

```

public static boolean[][] readTextFile(String dosyaAdi)
throws FileNotFoundException, IOException{

    boolean[][] arrayCiphertext = new boolean[100][64];
    BufferedReader oku = new BufferedReader(new
    FileReader(dosyaAdi));
    String satir = oku.readLine();
    int cipherTextIndex = 0;

    while (satir != null)
    {   String[] parts = satir.split(",");
        for (int i = 0; i < 64; i++)
        {
            arrayCiphertext[cipherTextIndex][i]
            = Boolean.parseBoolean(parts[i]);
        }
        cipherTextIndex++;
        satir = oku.readLine();
    }

    return arrayCiphertext;
}

public static boolean[] Encryption(boolean[] key, boolean[] plaintext){

    boolean[] postAddKeyMatrix = addKeyOperation(key, plaintext);
    boolean[] postSubstitutionMatrix
    = SubstitutionOperation(postAddKeyMatrix);
    boolean[] shiftResultMatrix = new boolean[64];
    ShiftOperation(postSubstitutionMatrix, shiftResultMatrix, 0, 0);
    ShiftOperation(postSubstitutionMatrix, shiftResultMatrix, 1, 1);
    ShiftOperation(postSubstitutionMatrix, shiftResultMatrix, 2, 12);
}

```

```

        ShiftOperation(postSubstitutionMatrix, shiftResultMatrix, 3, 13);

        return shiftResultMatrix;
    }

    public static boolean[] Decryption(boolean[] key1, boolean[] key2,
        boolean[] ciphertext){

        boolean[] shiftResultMatrix = new boolean[64];
        ShiftOperation(ciphertext, shiftResultMatrix, 0, 0);
        ShiftOperation(ciphertext, shiftResultMatrix, 1, 15);
        ShiftOperation(ciphertext, shiftResultMatrix, 2, 4);
        ShiftOperation(ciphertext, shiftResultMatrix, 3, 3);

        boolean[] postSubstitutionMatrix =
            SubstitutionOperationInverse(shiftResultMatrix);

        boolean[] postAddKeyMatrix =
            addKeyOperation(key2, postSubstitutionMatrix);

        ShiftOperation(postAddKeyMatrix, shiftResultMatrix, 0, 0);
        ShiftOperation(postAddKeyMatrix, shiftResultMatrix, 1, 15);
        ShiftOperation(postAddKeyMatrix, shiftResultMatrix, 2, 4);
        ShiftOperation(postAddKeyMatrix, shiftResultMatrix, 3, 3);

        postSubstitutionMatrix =
            SubstitutionOperationInverse(shiftResultMatrix);
        postAddKeyMatrix = addKeyOperation(key1, postSubstitutionMatrix);

        return postAddKeyMatrix;
    }

```

```

public static void PrintMatrix(String header1,String header2,
String header3, boolean[] values1, boolean[] values2,
boolean[] values3)
{

    System.out.println("  "+ header1 +"  "
+ header2 + "  " + header3);

    for (int r = 0; r < 4; r++)
    {   String rowStr1 = "";
        String rowStr2 = "";
        String rowStr3 = "";

        for (int c = 15; c >= 0; c--)
        {
            int linearIndex = r * 16 + c;
            rowStr1 += (values1[linearIndex] == true ? "1" : "0");
            rowStr2 += (values2[linearIndex] == true ? "1" : "0");
            rowStr3 += (values3[linearIndex] == true ? "1" : "0");
        }
        System.out.println(rowStr1 + "  " + rowStr2 + "  "
+ rowStr3);
    }
    System.out.println("");
}

```

```

public static void ShiftOperation(boolean[] input, boolean[] output,
int rowIndex, int shiftAmount) {

    int maxIndex = (rowIndex + 1) * 16;

    for (int i = rowIndex * 16; i < maxIndex; i++)
    {

```

```

        int newIndex = i + shiftAmount;
        if (newIndex >= maxIndex) {
            newIndex -= 16;
        }
        output[newIndex] = input[i];
    }
}

```

```

public static int CheckCounter(boolean[] ciphertext1, boolean[]
ciphertext2){

```

```

    int counter=0;

    if( ciphertext1.equals(ciphertext2)){
        counter++;
    }

    return counter;
}

```

```

public static boolean[] SubstitutionOperation(boolean[] postAddKeyMatrix)
{

```

```

    boolean[] substitutionResultMatrix =
    new boolean[postAddKeyMatrix.length];

    for (int c = 0; c < 16; c++) {
        int inputHexValue = 0;
        inputHexValue += postAddKeyMatrix[c + 0] == true ? 1 : 0;
        inputHexValue += postAddKeyMatrix[c + 16] == true ? 2 : 0;
        inputHexValue += postAddKeyMatrix[c + 32] == true ? 4 : 0;
        inputHexValue += postAddKeyMatrix[c + 48] == true ? 8 : 0;
    }
}

```

```

    int outputHexValue = SBoxOperation(inputHexValue);
    substitutionResultMatrix[c + 0] =
        ((outputHexValue & 1) == 0) ? false : true;
    substitutionResultMatrix[c + 16] =
        ((outputHexValue & 2) == 0) ? false : true;
    substitutionResultMatrix[c + 32] =
        ((outputHexValue & 4) == 0) ? false : true;
    substitutionResultMatrix[c + 48] =
        ((outputHexValue & 8) == 0) ? false : true;
}

return substitutionResultMatrix;
}

public static boolean[] SubstitutionOperationInverse(boolean[]
postAddKeyMatrix) {

    boolean[] substitutionResultMatrix =
new boolean[postAddKeyMatrix.length];

    for (int c = 0; c < 16; c++) {
        int inputHexValue = 0;
        inputHexValue += postAddKeyMatrix[c + 0] == true ? 1 : 0;
        inputHexValue += postAddKeyMatrix[c + 16] == true ? 2 : 0;
        inputHexValue += postAddKeyMatrix[c + 32] == true ? 4 : 0;
        inputHexValue += postAddKeyMatrix[c + 48] == true ? 8 : 0;

        int outputHexValue = SBoxOperationInverse(inputHexValue);
        substitutionResultMatrix[c + 0] =
            ((outputHexValue & 1) == 0) ? false : true;
        substitutionResultMatrix[c + 16] =
            ((outputHexValue & 2) == 0) ? false : true;
        substitutionResultMatrix[c + 32] =

```

```

        ((outputHexValue & 4) == 0) ? false : true;
        substitutionResultMatrix[c + 48] =
        ((outputHexValue & 8) == 0) ? false : true;
    }

    return substitutionResultMatrix;
}

public static boolean[] addKeyOperation(boolean[] key,
boolean[] plaintext) {

    boolean output[] = new boolean[64];
    for (int i = 0; i < 64; i++) {
        output[i] = key[i] ^ plaintext[i];
    }

    return output;
}

public static int SBoxOperation(int input) {

    int[] sBox = {9, 4, 15, 10, 14, 1, 0, 6, 12, 7, 3,
    8, 2, 11, 5, 13};

    return sBox[input];
}

public static int SBoxOperationInverse(int input) {

    int[] sBox = {6, 5, 12, 10, 1, 14, 7, 9, 11, 0, 3,
    13, 8, 15, 4, 2};

    return sBox[input];
}

```



```

}

public static void PrintMatrix2(String header, boolean[] values)
{

    System.out.println("");
    System.out.println(" * " + header);
    System.out.println("=====");

    for (int r = 0; r < 4; r++)
    {
        String rowStr = "";
        for (int c = 15; c >= 0; c--)
        {
            int linearIndex = r * 16 + c;
            rowStr += (values[linearIndex] == true ? "1" : "0") + " ";
        }
        System.out.println(rowStr);
    }
    System.out.println("");
}
}

```