SLIDE ATTACK AND ITS APPLICATIONS

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

BY

ERKAN USLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CRYPTOGRAPHY, METU

SEPTEMBER 2017

Approval of the thesis:

SLIDE ATTACK AND ITS APPLICATIONS

submitted by **ERKAN USLU** in partial fulfillment of the requirements for the degree of **Master of Science in Department of Cryptography, METU, Middle East Technical University** by,

Prof. Dr. Bülent Karasözen
Director, Graduate School of **Applied Mathematics** _____

Prof. Dr. Ferruh Özbudak
Head of Department, **Cryptography, METU** _____

Assoc. Prof. Dr. Ali Doğanaksoy
Supervisor, **Department of Mathematics, METU** _____

Dr. Muhiddin Uğuz
Co-supervisor, **Department of Mathematics, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Murat Cenk
Cryptography, METU _____

Prof. Dr. Ersan Akyıldız
Department of Mathematics, METU _____

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

Assist. Prof. Dr. Fatih Sulak
Department of Mathematics, Atılım University _____

Dr. Muhiddin Uğuz
Department of Mathematics, METU _____

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: ERKAN USLU

Signature :

ABSTRACT

SLIDE ATTACK AND ITS APPLICATIONS

Uslu, Erkan
M.S., Department of Cryptography, METU
Supervisor : Assoc. Prof. Dr. Ali Doğanaksoy
Co-Supervisor : Dr. Muhiddin Uğuz

September 2017, 39 pages

Block ciphers, widely used in cryptography, have been designed to encrypt large amount of data such as public sector services, banking services, Healthcare contributions. With the increment of technological developments, they have also been started to be used for small data in industrial products such as Internet of Things, smart cards, car keys etc. These types of cryptosystems are called as lightweight cryptosystems. Similar to other cryptographic algorithms, the ones used in lightweight systems need to be tested towards cryptanalytic techniques. The most common techniques are differential and linear cryptanalysis. However, they become less efficient when the number of rounds in algorithms is increased. At this point, a new method called slide attack which is independent of the number of rounds is developed.

This thesis focuses on the fundamentals of the slide attack and especially how it works on block ciphers. Additionally, some applications that will be beneficial to understand slide attack is given. Moreover, we give a practical attack to a variant of PRESENT lightweight block cipher.

Keywords : Slide Attack, Block Ciphers, Lightweight Cryptography, Cryptanalysis

ÖZ

SLIDE ATAK VE UYGULAMALARI

Uslu, Erkan
Yüksek Lisans, Kriptografi Bölümü
Tez Yöneticisi : Doç. Dr. Ali Doğanaksoy
Ortak Tez Yöneticisi : Dr. Muhiddin Uğuz

Eylül 2017, 39 sayfa

Blok şifreler, büyük boyuttaki verilerin şifrelenmesinde kullanılan bir şifreleme yöntemidir. Bu verilere örnek olarak, kamusal bilgiler, bankacılık işlemlerinde kullanılan bilgiler, sağlık sektöründe kullanılan bilgiler örnek olarak gösterilerbilir. Teknolojik gelişmeler neticesinde blok şifrelere küçük boyuttaki verilerin şifrelenmesinde de ihtiyaç duyulmaktadır. Örnek olarak, akıllı kartlarda, araba anahtarlarında, kısacası nesnelerin interneti olarak adlandırılan yapıların büyük bir kısmında blok şifreler kullanılmaktadır. Bu tür şifreleme sistemlerine, hafif siklet kriptosistemler denmektedir. Blok Şifrelere yapılan en ünlü ataklar lineer ve diferansiyel kriptanaliz yöntemleridir. Fakat hafif siklet algoritmalar bu ataklardan etkilenmemektedir. Çünkü hafif siklet algoritmaların güvenliği, dizayn kriterleri ile birlikte tur sayısının çok fazla olmasıdır. Yine de bu algoritmaların zayıf olduğu kriptanaliz yöntemleri bulunmaktadır. Kaydırma atakları bunlardan biridir. Kaydırma atakları tur sayısından bağımsız olarak uygulanmaktadır. Bu nedenle özellikle hafif siklet algoritmalar için önemli bir tehdit oluşturmaktadır.

Bu tezde kaydırma ataklarının temellerinden ve matematiksel altyapılarından bahsedilecektir. Ayrıca kaydırma ataklarını daha iyi anlamak için örnek uygulamalar anlatılmaktadır. Buna ek olarak bir hafif siklet algoritması olan PRESENT'in sadeleştirilmiş versiyonuna bir atak önerilmiştir.

Anahtar Kelimeler : Kaydırma Atakları, Blok Şifreler, Kriptanaliz, Hafif Siklet Kriptosistemler

Dedicated to My Father, My Mother, My Brother and My Nephew

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor Assoc. Prof. Dr. Ali Doğanaksoy for the useful comments, remarks and engagement through the learning process of this master thesis.

Furthermore, I would like to thank my co-supervisor Dr. Muhiddin Uğuz. The door to his office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work, but steered me in the right the direction whenever he thought I needed it.

In addition, I would like to thank Assist. Prof. Dr. Fatih Sulak and Dr. Onur Koçak for their advices to improve the content of this thesis.

I would like to express my special gratitude and thanks to the members of "Aknaz Reading Group" for imparting their knowledge, guidance and expertise in this thesis.

Finally, I must express my very profound gratitude to my parents and to my childhood and lifelong friends Gülseda Uz, Onur Koşar, Faruk Sevgili, Mustafacan Kutsal, Çağatay Karakan, Yiğit Şiar Başol, Gözde Erginbay, Rasim Hüseyin Baloğlu, Hazal Şahin, Barış Pekçağlayan, Gökhan İpekkan for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ABBRV	Abbreviation
$\mathbb{R}$	Real Numbers

CHAPTER 1

INTRODUCTION

1.1 Cryptology

Main concern of cryptology is security of communication and storage of data. It uses theories and techniques from mathematics, statistics, computer science and electronic engineering. Its aim is to hide the information from all but those who are authorised. Moreover, some topics of cyber security such as authentication, confidentiality, data integrity and non-repudiation are provided by cryptologic structures.

Cryptography is the techniques that can be used to provide the security of communication from opponents. Cryptanalysis, on the other hand, is the works of analysing the security of cryptographic algorithms. These have been in interaction with each other during the period of their evolution. As soon as a new technique is developed by cryptanalysists to break an algorithm, cryptographers improve their algorithms to make them secure against that newly developed attack if possible, as otherwise those algorithms become useless. The more computation power the computers have, the more brute force attacks there are. Hence, cryptographers have to improve their encryption algorithms continually and develop new encryption techniques.

1.2 Description of Cryptography

There are two main encryption approaches in modern cryptography called symmetric key cryptography and asymmetric key cryptography. Generally, a symmetric encryption algorithm uses the same key for both encryption and decryption. This key should be shared between the sender and the receiver. On the other hand, asymmetric encryption algorithm makes use of two keys called private key and public key. These key pairs are produced together using some mathematically hard problems. The basic idea behind this system is that each user shares his or her public key with everyone, but keeps the private key as secret. Anyone can send a message to anyone by encrypting the message with that person's public key. However, it is almost impossible to decrypt a secret message without having receiver's private key. Although these types of algorithms are slower and require more computation power compared to symmetric algorithms, the main advantage of this type of encryption is that parties do not need to come together

to share a private key. When communication with symmetric algorithm is preferred, the secret key can be shared to other parties by means of an asymmetric algorithm at the first place. This is called key exchange. However, asymmetric cryptosystems will not be our concern in this thesis.

1.2.1 Stream Ciphers

Stream ciphers being a branch of symmetric encryption, encrypt a plaintext digit at a one-time combining with a key-stream digit in order to obtain a digit of ciphertext. This operation repeats until the whole ciphertext digits are produced. To perform this process, keystream needs to be as long as the plaintext. For this, there are different methods to produce a long and random looking keystreams from a short key.

1.2.2 Block Ciphers

Block ciphers being the second branch of symmetric encryption, encrypt large amount of data block by block according to its designed criteria, such as encryption routines, number of rounds, protocols, etc. Generally, block ciphers have one secret key. This key is used during the encryption routine with a key generation algorithm which generates round keys. When the rounds of the encryption algorithm are successively applied to the plaintext the ciphertext is obtained. Decryption routine is just the inverse of this process.

Block ciphers can be categorised mainly as Substitution - Permutation Networks and Feistel Structures.

1.3 Cryptanalysis of Block Ciphers

In cryptology, encryption algorithm is assumed to be public according to Kerckhoffs's Principle. The only secret information is the key. The main technique to increase the security of an algorithm is to apply it repeatedly with different round keys. That is, the number of rounds of encryption is determined by how many times the round functions should be repeated. To produce the key for each round, key schedule is applied by using the master key. The security of the algorithm must depend only on the key.

The generic attack scenario to block ciphers is that the attacker tries every possible n-bit keys on a known plaintext-ciphertext pair in order to find the true key. This attack is called as Brute Force or Exhaustive Search. To make Brute Force infeasible, designers of a block cipher tend to use large key spaces in a block cipher.

There are different attack scenarios with different assumptions:

- **Ciphertext-only attack:** It is assumed that an attacker only has encrypted text.

If the attacker can obtain corresponding plaintext or encryption key somehow, she/he is considered successful.

- **Known-plaintext attack:** The attacker has some plaintexts and corresponding ciphertexts. These plaintext - ciphertext pairs are assumed to be random. This method is aimed to capture the key using the encryption.
- **Chosen-plaintext attack:** In this case, the attacker affects some plaintext bits to observe that how ciphertext bits are changed. This information is used for capture the some(or whole) key bits.
- **Chosen-ciphertext attack:** The attacker has an authorisation for decrypt chosen ciphertexts in order to obtain corresponding plaintexts.
- **Adaptive chosen plaintext/ciphertext attack:** The attacker is authorised for encryption or decryption. Depending on his expectation from the characteristics of plaintexts/ciphertexts after encryption/decryption process, the attacker can do this process repeatedly.

1.3.1 Complexity

In order to mount a reasonable attack, the attacker should consider that how many data and how much time she/he needs.

- **Time Complexity** is the entire time of an algorithm from start until its finalization. Generally, it is obtained by calculating the number of encryption executed.
- **Data Complexity** is the measure of plaintext-ciphertext pairs needed to perform an attack.
- **Memory Complexity** is the total amount of storage required to apply an attack.

1.3.2 Linear and Differential Cryptanalysis

Linear[15] and differential cryptanalysis[2, 3] are the two important attack methods to recover the key of block ciphers. Basically, the idea of linear cryptanalysis is to find linear approximations over the non-linear operations of the cipher. Whereas, differential cryptanalysis is based on the probability of a pair of input and output occurrence of difference.

1.4 About this Thesis

Since it is usually considered to be sufficient to increase the number of rounds for enhancing the security, some researchers head towards other methods to change this idea. In this sense, a successful attack mechanism, which is named as Slide Attack[4],

was presented by Alex Biryukov and David Wagner in 1999. Basically, the attack works on the algorithms that have weak round functions and weak key schedules. The main characteristic of slide attack is independent of the number of rounds. For some algorithms, it has been shown that slide attack is an extremely effective method to decrease the time and the data complexities.

The organisation of this thesis as follows. In Chapter 2, brief preliminaries about this thesis are given. In Chapter 3, mathematical backgrounds and fundamentals of slide attack are explained. Chapter 3 also includes some generic techniques to apply the slide attack over some kind of block cipher structures. In Chapter 4, there are some applications of slide attack. In Chapter 5, the conclusion is given.

CHAPTER 2

PRELIMINARIES

In this chapter, preliminaries of slide attack will be given. Firstly, two basic types of block cipher algorithms are discussed with examples. Then, lightweight cryptography which is designed for low cost environment are given with two sample algorithms; PRESENT[8] and KeeLoq[10]. Finally, an important concept in computing complexity of slide attack called Birthday Paradox[16] will be explained in details.

2.1 Block Ciphers

As mentioned in Chapter 1, a block cipher partitions the plaintext into fixed-size blocks and encrypts one block at a time. The encrypting process is applied in iterated round functions.

Naturally, block ciphers have many different design criteria. However, generally, they are categorised as Substitution-Permutation Networks[14] and Feistel Networks[17].

2.1.1 Substitution-Permutation Networks(SPNs)

One of the most important block cipher method is Substitution-Permutation Network[14]. Basically, SPN's takes a plaintext and a key as input. First, plaintext block is XORed with round-key that is produced by key schedule. This operation is called Round 0. After that, the output is partitioned into small blocks and each part is passed through a nonlinear layer to satisfy confusion called S-Box. Moreover, the output bits are permuted in order to satisfy the diffusion property with the help of permutation boxes and xored with the round key. This combination is called a round. These rounds are applied repeatedly and finally, the ciphertext is obtained.

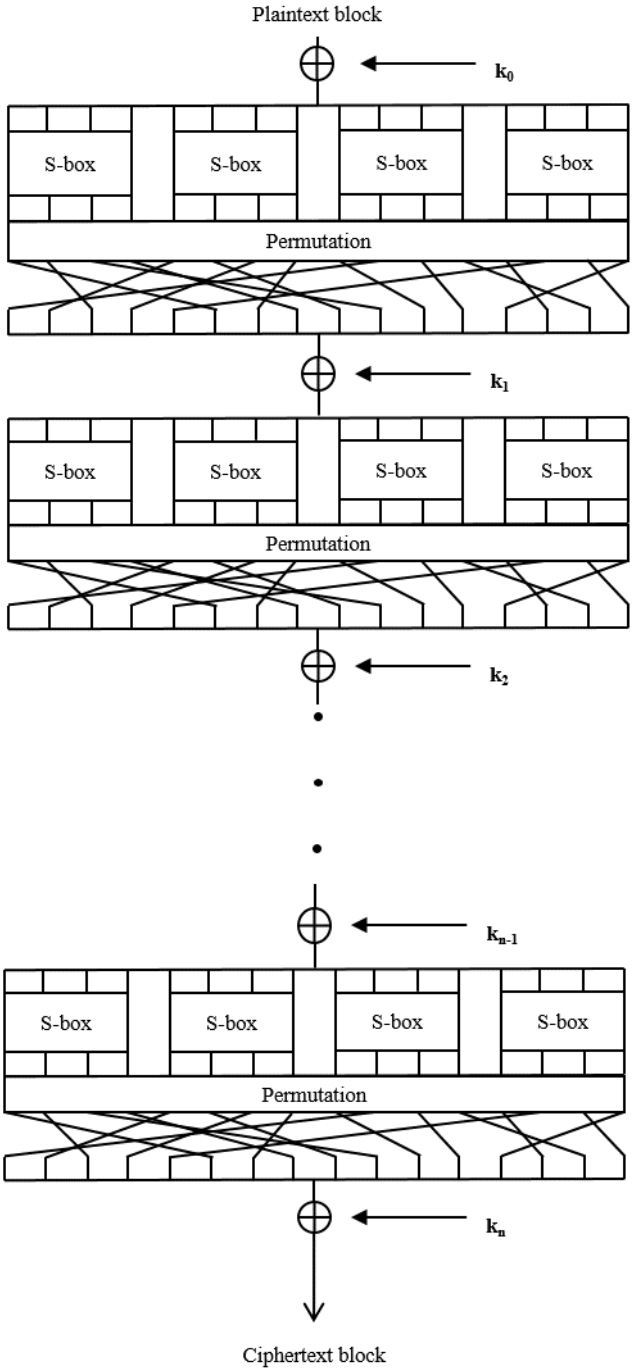

Figure 2.1: Substitution - Permutation Network

2.1.1.1 Advanced Encryption Standard(AES)

AES[9],[18] is the most popular example for the Substitution - Permutation Network among block ciphers in cryptography. It is a variant of a Rijndael Cipher produced by Belgian Cryptographers Vincent Rijmen and Joan Daemen. This variant was the winner of NIST's competition so that it is called as Advanced Encryption Standard(AES).

There are three variant of AES depending on the key size, that are 128/196/256 bits and the number of rounds that are 10/12/14 respectively. Each version encrypts 128 bits plaintext blocks. First, there is an additional AddRoundKey operation before the first round. Each round has the same structure that is combination of SubByte(SB), ShiftRows(SR), MixColumn(MC) and AddRoundKey(ARK). Besides, the round key is produced by key schedule algorithm in every round. Note that, MixColumn operation is ignored in the last round. Encryption process of AES is shown in Figure 2.2

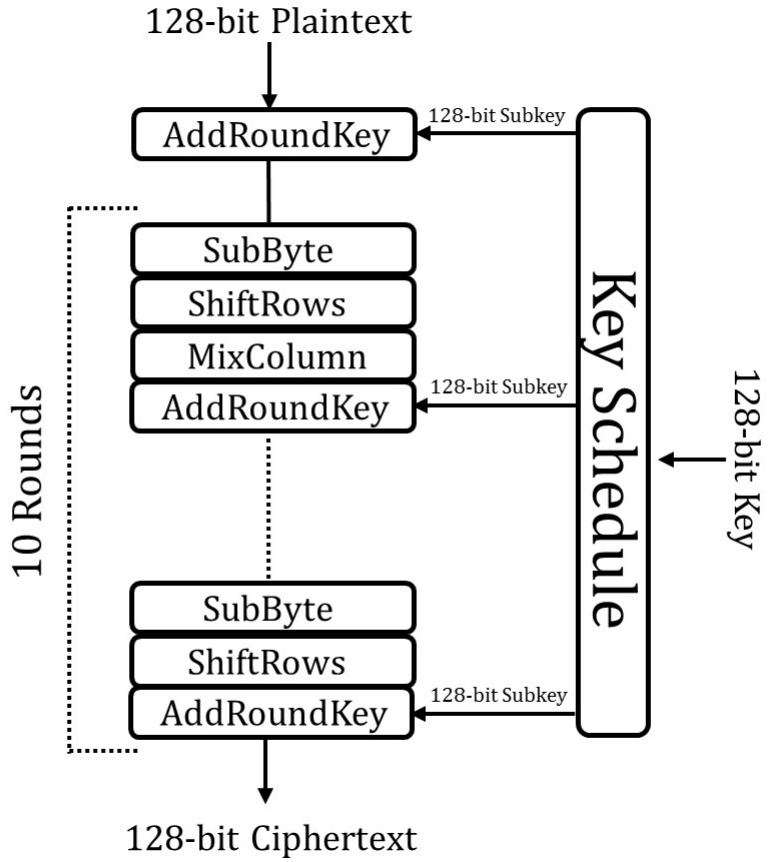

Figure 2.2: Advanced Encryption Standard

2.1.2 Feistel Structure

Feistel Structure is the second most popular method for block ciphers. It was invented by German Cryptographer Horst Feistel when he was working for IBM (USA). There are too many block ciphers using this structure, such as DES, Blowfish, TEA etc. In a Feistel Structure, encryption and decryption routines are very similar, yet sometimes identical. Generally, there is a key schedule algorithm which produces round-keys in order to be used in every round function F, called Feistel Function. The plaintext P_i is split into two equal-length parts L_i and R_i. Every i-th round, the following operations are performed:

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

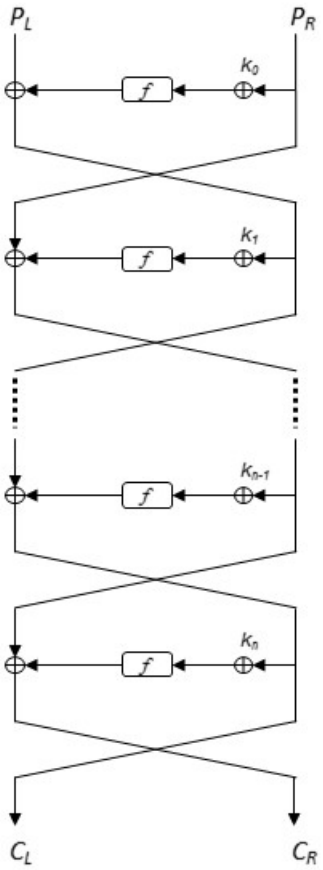

Figure 2.3: Feistel Structure

2.1.2.1 Data Encryption Standard(DES)

Data Encryption Standard[17] is the most popular instance for Feistel Networks. First, it was designed with the name Lucifer[11][12] by Horst Feistel. Afterwards, it was

slightly modified by National Security Agency (NSA) and published with the name Data Encryption Standard (DES).

DES uses 56-bit master key. However, the round keys are 48-bits which are produced from this key. The length of the plaintext is 64-bit. In every round, the 32-bit half block of plaintext goes through to the Feistel Function. In the Feistel Function, the input is expanded to 48-bit and then XORed with 48-bit round key. After that, the value is substituted and compressed to 32-bit by S-boxes. Finally, it is permuted go out from Feistel Function. The output of Feistel Function is XORed with the other half of plaintext and the halves are swapped before the next round. This routine is iterated 16-times. Encryption scheme of DES is shown in Figure 2.4

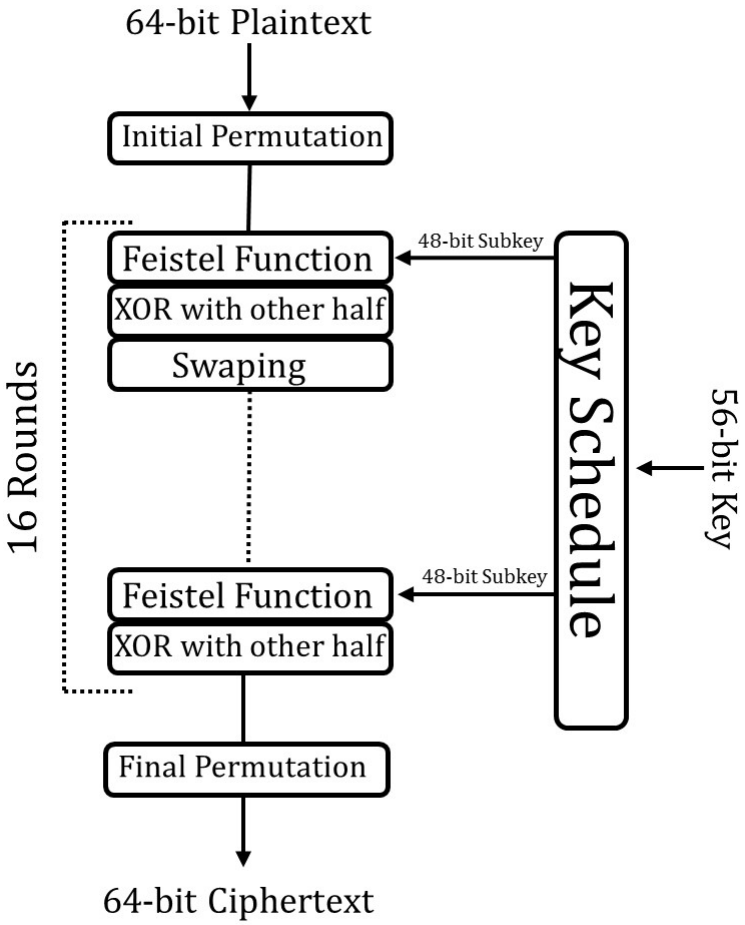

Figure 2.4: Data Encryption Standard

2.2 Lightweight Cryptosystems

Block ciphers, like DES and AES are used for enciphering big amount of data(i.e. informations about a company, hard discs etc.) and this leads to a high cost require-

ments such large storage and high CPU power. However, improvement of microchips leads to show the necessity for new encryption algorithms which work with low costs. These algorithms are named as Lightweight Cryptosystems. Generally, lightweight algorithms are also block ciphers. Because of the low-cost requirements, these kinds of ciphers use primitive key generation algorithms and small number of non-linear truth tables. These structures of lightweight systems seem more insecure. However, these algorithms run with high number of rounds and this provides basic security requirement for lightweight systems against linear and differential cryptanalysis.

2.2.1 PRESENT

PRESENT[8] was introduced by Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Axel Poschmann ,Yannick Seurin, Matthew J. B. Robshaw, Christof Paar, and C. Vikkelsoe. in 2007. It is also known as Ultra-Lightweight Block Cipher.

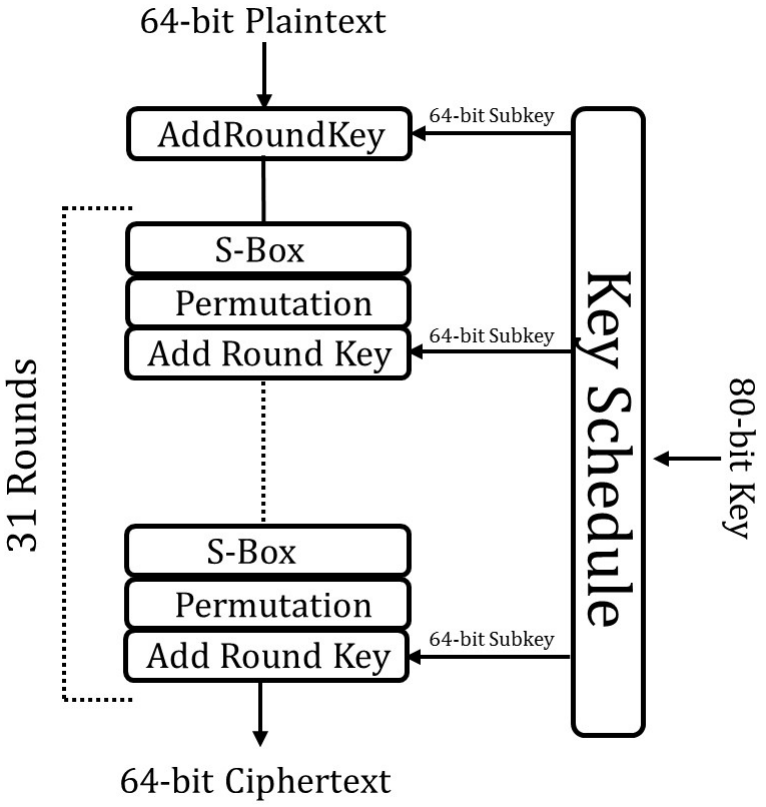

Figure 2.5: PRESENT Encryption

PRESENT is a lightweight block cipher based upon Substitution-Permutation Network. It consists of 31-rounds with 64-bit plaintext block size. There are two key size options, 80-bit and 128-bit that can be used in the cipher. Encryption process is exactly the same with classical SPN. In every round, 64-bit plaintext is XORed with the round key. After XOR operation, the output is passed through S-Boxes and permuta-

tion layer. After 31 rounds iteration, the output is XORed with round key in order to obtain the ciphertext. Encryption scheme is as Figure 2.5

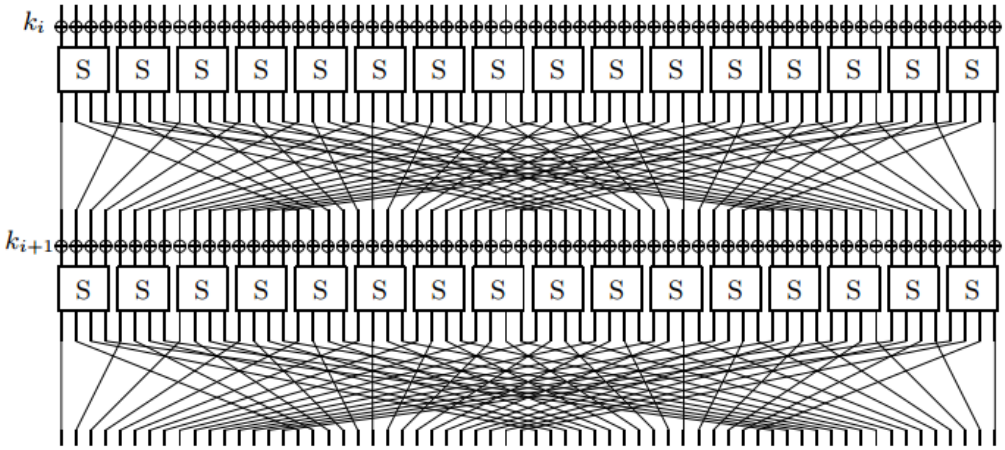

Figure 2.6: The round function of PRESENT

Round-keys are produced from a master key that can be 80-bit or 128-bit. We explain only 80-bit provided key scheduling algorithm in this section. The user-provided 80-bit secret key is stored as $K = k_{79}k_{78}...k_0$. The i-th round key K_i is the left-most significant 64-bit of the K. When K_i is obtained, K is rotated to left by 61 bit-times. Then, the left-most significant 4-bit goes through S-box layer. Finally, the key block $k_{19}k_{18}k_{17}k_{16}k_{15}$ of K is XORed with the round counter value which is determined uniquely for each round.

2.2.2 KeeLoq Block Cipher

KeeLoq[10] is a lightweight block cipher which is developed by Microchip Technology Inc. to be used in wireless authentication applications such as keyless entry systems, car authentication systems etc. It provides secure authentication with power efficient hardware implementations for very low costs. This made KeeLoq once-preferable cipher of its field until the first cryptanalysis[6] is published by Bogdanov in February 2007.

2.2.2.1 Encryption of KeeLoq

KeeLoq is a lightweight cryptosystem that uses a 64-bit key with 32-bit blocks. It has 528 iterative rounds and one key-bit is used in each round operation. Every round operation of KeeLoq is equivalent to a repetition of a non-linear feedback shift register (NLFSR). To be more specific, using the notation of [13], let

$$Y^{(i)} = (y_{31}^{(i)}, ..., y_0^{(i)}) \in \{0,1\}^{32} \tag{2.1}$$

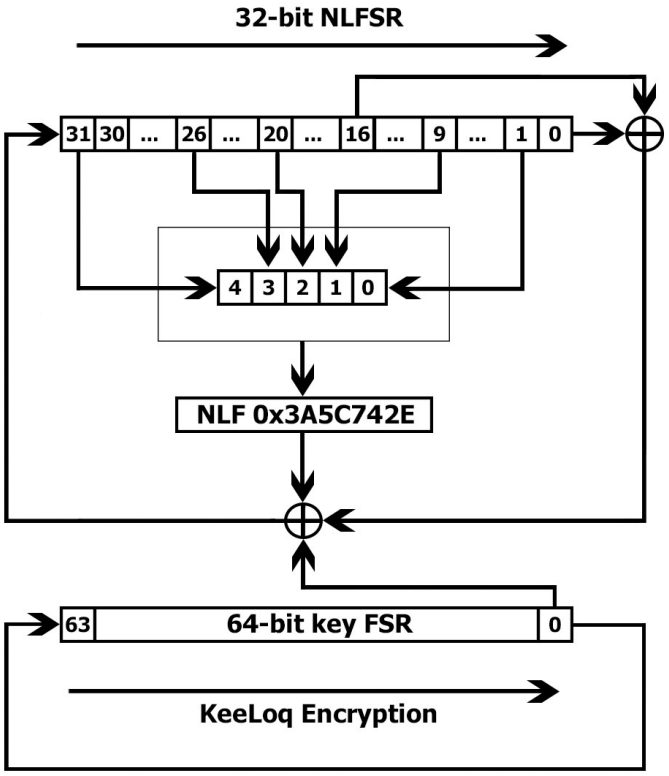

Figure 2.7: Encryption routine of KeeLoq

be the input of round i where $i \in [0, 528)$, and let

$$K = (k_{63}, ..., k_0) \in \{0, 1\}^{64} \tag{2.2}$$

be the key. The plaintext, $Y^{(0)} = P$, is the input of round 0. The ciphertext, $C = Y^{(528)}$, is the output after 528 rounds. The round function can be described as follows:

$$\phi^{(i)} = NLF(y_{31}^{(i)}, y_{26}^{(i)}, y_{20}^{(i)}, y_{9}^{(i)}, y_{1}^{(i)}) \oplus y_{16}^{(i)} \oplus y_{0}^{(i)} \oplus k_{i(mod64)} \tag{2.3}$$

$$Y^{(i+1)} = (\phi^{(i)}, y_{31}^{(i)}, ..., y_{1}^{(i)}) \tag{2.4}$$

We can represent the non-linear function in its algebraic normal form (ANF) as:

$$\begin{aligned} &NLF(x_4, x_3, x_2, x_1, x_0) = x_4x_3x_2 \oplus x_4x_3x_1 \oplus x_4x_2x_0 \oplus x_4x_1x_0 \\ &\oplus x_4x_2 \oplus x_4x_0 \oplus x_3x_2 \oplus x_3x_0 \oplus x_2x_1 \oplus x_1x_0 \oplus x_1 \oplus x_0 \end{aligned} \tag{2.5}$$

NLF is a boolean function of 5 variables with output vector $3A5C742E_x$. In other words, $NLF(i)$ is the i-th bit of this hexadecimal constant, where $i = 16x_4 + 8x_3 + 4x_2 + 2x_1 + x_0$. For instance, $(0, 0, 1, 0, 1)$ gives $i = 5$ and the fifth least significant (sixth one from the right) bit of $(00111010010111000111010000101110) = 3A5C742E_x$. For further information can be examined.

The inverse round function is used in decryption as follows:

i ranges from 528 down to 1;

$$\theta^{(i)} = NLF(y_{30}^{(i)}, y_{25}^{(i)}, y_{19}^{(i)}, y_{8}^{(i)}, y_{0}^{(i)}) \oplus y_{15}^{(i)} \oplus y_{31}^{(i)} \oplus k_{i-1(mod64)} \tag{2.6}$$

$$Y^{(i-1)} = (y_{30}^{(i)}, ..., y_{0}^{(i)}, \theta^{(i)}) \tag{2.7}$$

2.2.2.2 The IFF Protocol based on KeeLoq

There are two known protocols supported by KeeLoq, the "Code Hopping" and "Identify Friend or Foe (IFF)". The latter one offers an opportunity to collect required data i.e. plaintext-ciphertext pairs to perform slide attack with respect to the former. This collection can be gathered with a transponder in the range of the encoder. Therefore, the only IFF protocol is mentioned in the following passage. IFF, which is mostly used in car authentication, is a challenge-response protocol between the partners, the encoder (e.g. the car key) and the decoder (e.g. the car). It begins with a 32-bit challenge being sent from the decoder to the encoder. When the encoder gets this challenge, it encrypts with KeeLoq and the shared key and sends the ciphertext back to the decoder. Then knowing the encryption of the challenge, the decoder verifies whether the ciphertext comes from the encoder or not. If so, the ciphertexts are equal, the decoder unlocks the car. If this is not the case, it activates the alarm due to the failure of authentication. Note that in this protocol, the encoder activates with the signal i.e. challenge coming from the decoder and the challenge is automatically presented by an incidence of the encoder. Therefore, there is no need for either a battery or pressing of a button on the encoder.

2.3 Birthday Paradox

Birthday Paradox[16] explains that from a collection of n different objects if $\sqrt{n}$-many are chosen randomly, there will be a collision with high probability $(P > 1/2)$. In slide attack, we need *slid pairs* that are as will be explained in Chapter 3 a kind of collision between plaintexts and ciphertexts. Therefore, it is useful for our work.

2.3.1 Outline of the Birthday Paradox

Let $P(k)$ denote the probability that randomly chosen k objects from a collection of n-different object, with replacement, are all different. Then by the Pigeonhole Principle, $P(k) = 0$ if $k > n$. We want to prove that $P(k) > \frac{1}{2}$ when $k > \sqrt{n}$.

Note that,

$P(k) = 1 \cdot (1 - \frac{1}{n}) \cdot (1 - \frac{2}{n}) \cdots (1 - \frac{k-1}{n})$

Using Maclaurin expansion of e^x, we can write;

$e^x = \sum_{r=0}^{\infty} \frac{x^r}{r!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} \cdots$

Hence we can use the following approximation;

$e^x \approx 1 + x$ and consequently $e^{-x} \approx 1 - x$.

Using this approximation, we can approximate $P(k)$ as follows;

$P(k) \approx 1 \cdot e^{-1/n} \cdot e^{-2/n} \cdots e^{-(k-1)/n} = e^{\frac{-1\cdot(1+2+\cdots+(k-1))}{n}} = e^{\frac{-k\cdot(k-1)}{2n}} \approx e^{\frac{-k^2}{2n}}$.

We want to find k so that there is a crash with probability at least equal to $\frac{1}{2}$. In other words; find the smallest value of k so that $P(r) < \frac{1}{2} \quad \forall r \geq k$.

Thus we need to solve the following inequality for k;

$$e^{\frac{-k^2}{2n}} < \frac{1}{2}$$

Taking natural logarithm of both sides, we obtain;

$$-\frac{k^2}{2n} < -\ln 2 \Rightarrow \frac{k^2}{2n} > \ln 2$$

$$\Rightarrow k^2 > 2\ln 2 \cdot n \Rightarrow k > \sqrt{2\ln 2 \cdot n} \approx \sqrt{n}$$

CHAPTER 3

SLIDE ATTACK

Slide Attack[4],[5] is a cryptanalytic method applied to block ciphers. Increasing the number of rounds of a block cipher usually increases the security level especially for the linear and differential attacks. In contrast to those attacks, slide attack is independent of the number of rounds of a block cipher when applicable. In this thesis our aim is to delve deeper into the slide attack method and expand on it in a mathematical sense . In this section, we explain how slide attack works in details.

Initially, an iterative block cipher with r-rounds is considered. To perform the attack, the cipher should have a weak key schedule. In the standard scenario, it is assumed that the encryption function of a cipher E is a composition of identical key-dependent permutation F in notation $E : F \circ F \circ ... \circ F = F^s$ where each F can be one or more than one round of the cipher. It is further assumed that F is weak in the sense that if two pairs of input-output of F is known, in other words, if

$$\begin{aligned} F_k(x_1) &= y_1, \\ F_k(x_2) &= y_2, \end{aligned} \tag{3.1}$$

are known, then the key k in use can be recovered easily.

For such constructions, encryption function is denoted by $E_k : F_k^s : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$, where F_k corresponds to r-rounds of encryption E_k which in total $r.s$ rounds, and n denotes the block size. Breaking such a cipher means obtaining the key used in F_k. The idea is that in the collection of all plaintext-ciphertext pairs, if one can to find a pair (called slid pair) such that;

$$\begin{aligned} &F_k(P_i) = P_j \text{ and hence,} \\ &F_k(C_i) = C_j \text{ where } E_k(P_i) = C_i \text{ and } E_k(P_j) = C_j, \end{aligned}$$

then by the weakness assumption on F, key can be recovered.

Definition 3.1. Let E_k be a block cipher $E_k = F_k^s$ of $c.s$ rounds, and let $C_i = E_k(P_i)$, $C_j = E_k(P_j)$. The pair (P_i, C_i) and (P_j, C_j) are called as a slid pair if $F_k(P_i) = P_j$ (and hence $F_k(C_i) = C_j$).

The above concept visually looks like, there are two encryption functions one of them is fixed, another is slid one or more round. Thus, this attack method is called as slide attack. Illustration of the attack is as below.

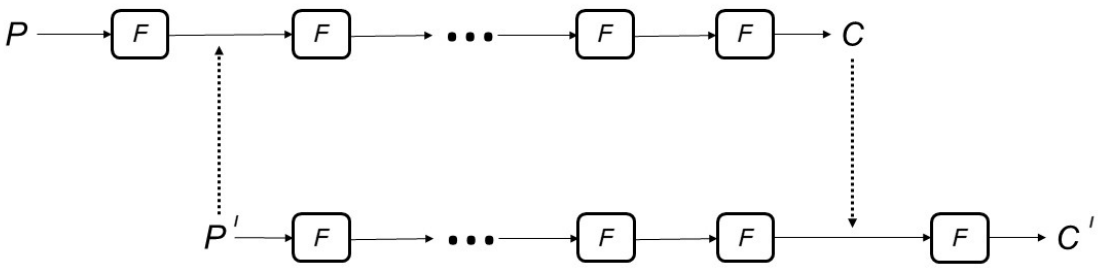

Figure 3.1: Illustration of Slide Attack

Notice that there are 2^n slid pairs in the collection of $2^n.2^n = 2^{2n}$ pairs $((P_i, C_i), (P_j, C_j))$. To find a slid pair, one needs to search through all 2^{2n} pairs until finding a satisfying the conditions of being slid pair. So, challenging yet attainable part of the attack is to find such pairs.

By Birthday Paradox, any collection of $\sqrt{2^n} = 2^{n/2}$ randomly chosen plaintext-ciphertext pairs, expectedly contain at least one slid pair. Thus, it comes out that slide attack is not useful since its data complexity is $2^{n/2}$ and time complexity is 2^n.

The main issue of the slide attack is to reduce its cost that is its complexity. Birthday Paradox is a suitable approach to reduce the data complexity. On the other hand, using properties of F, some filtering conditions can be defined in order to reduce the time complexity of this attack.

To reduce time complexity further, there are two alternative slide attack methods which are more effective than conventional slide attack called ***Complementation Slide*** and ***Sliding with a Twist***[5]. Generally, these are more effective to find a filtering condition when key schedule algorithm of a cipher produces non-identical round keys.

Complementation Slide Attack aims to find a filtering condition between slid pairs in order to perform an effective attack in terms of time complexity. If the key schedule of the algorithm produces two round keys, K_0 and K_1, differently, the difference named as slid difference $\Delta = K_0 \oplus K_1$ is introduced. This difference, between plaintext pairs, is preserved in all round operations appears as it is between the ciphertext pairs. In other words, following equations are observed:

$$\langle P'_L, P'_R \rangle = \langle P_R, P_L \oplus f(K_0 \oplus P_R) \rangle \oplus \langle \Delta, \Delta \rangle \tag{3.2}$$

$$\langle C'_L, C'_R \rangle = \langle C_R, C_L \oplus f(K_1 \oplus C_R \oplus \Delta) \rangle \oplus \langle \Delta, \Delta \rangle \tag{3.3}$$

Therefore, a $n/2$ bit filtering condition, $P'_L \oplus C'_L = P_R \oplus C_R$, is obtained from the above equations. When a slid pair is found, Δ can be extracted and all round keys can be found using the above equations.

Sliding with a Twist is another technique in order to find a filtering condition. In terms of Feistel Structures, encryption keys are inverse of the decryption keys. Assume K_0 and K_1 are round keys of encryption routine. For decryption, these keys are replaced like K_1 and K_0. Thus, there is a connection between slid pairs like that:

$$\langle C_L', C_R' \rangle = \langle P_L \oplus f(K_0 \oplus P_R), P_R \rangle \tag{3.4}$$

$$\langle P_L', P_R' \rangle = \langle C_L \oplus f(K_0 \oplus C_R), C_R \rangle \tag{3.5}$$

Therefore, the above equations gives a n bit filtering condition between slid pairs namely $C_R' = P_R$ and $P_R' = C_R$. As a result, one can find a slid pair with a hash table and can extract K_0. Finding a slid pair takes $2^{n/2}$ operation in the worst case scenario.

As a result, the main purpose of slide attack is to find slid pairs using the vulnerabilities of key scheduling algorithms and round functions of a block cipher in order to extract key bits independently from number of rounds. Sometimes, finding a slid pair takes more operation than brute force. In order to reduce complexities, one should catch some filtering conditions using weaknesses of structures or combining other crypt-analysis methods with slide attack. Below, there are block ciphering methods and descriptions of how slide attack is applied to them.

3.1 Slide Attack to Substitution-Permutation Networks

Usually, SPN's are more resistant than feistel structures to slide attack. However Biham and Dunkelman et al. give an efficient method to perform on SPN's with 1-round self similarity[1]. This attack aims to find a filtering condition so that it is required $\mathcal{O}(2^{n/2})$ data and $\mathcal{O}(2^{n/2})$ time in order to find the key.

3.1.1 Attack Scenario

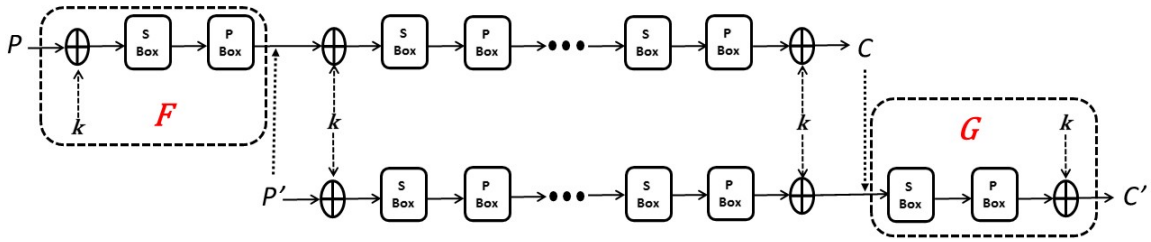

Figure 3.2: Slide Attack to SPN Ciphers

In Figure 3.2 it is shown that there is a different sliding approach in this method because of the presence of the key before the first round. Let P and P' forms a slid pair. As it can be seen in Figure 3.2, functions below can be determined:

$$F(P) = P' \text{ where } F(x) = Per(Sub(x \oplus K)) \text{ and } E(x) = F^r(x) \oplus K \tag{3.6}$$

$$G(C) = C' \text{ where } G(x) = Per(Sub(x)) \oplus K \text{ and } E(x) = G^r(x \oplus K) \tag{3.7}$$

If P and P' forms a slid pair then $F(P) = P'$ and $G(C) = C'$ should be satisfied. In other words;

$$\begin{aligned} &F(P) = P' \implies Per(Sub(P \oplus K)) = P' \implies P = Sub(Per(P')) \oplus K \\ &\text{Denoting } Sub(Per(P')) = \overline{P'}\text{,we have} \\ &P = \overline{P'} \oplus K \implies P \oplus \overline{P'} = K \end{aligned} \tag{3.8}$$

Additionally, G function has almost the same results as F:

$$\begin{aligned} &G(C) = C' \implies Per(Sub(C)) \oplus K = C' \\ &\text{Denoting} Per(Sub(C)) = \overline{C}\text{,we have} \\ &\overline{C} \oplus K = C' \implies \overline{C} \oplus C' = K \end{aligned} \tag{3.9}$$

After combining these two equations, we obtain a filtering condition:

$$P \oplus \overline{P'} = K = \overline{C} \oplus C' \implies P \oplus \overline{C} = \overline{P'} \oplus C' \tag{3.10}$$

This equality will be beneficial to reduce time complexity of finding slid pair. The examples of sliding method using these filtering condition are in Chapter 4. In [1], this method applied to 1K-AES which is a variant of AES block cipher. Additionally, we give attack to PRESENT lightweight block cipher using this technique.

3.2 Slide Attack to Feistel Structures

In some cases, slide attack is very effective method to cryptanalysis of the feistel structures. Basically, if (P_i, C_i) and (P_{i+1}, C_{i+1}) forms a slid pair, that is if $F(P_i) = P_{i+1}$ and hence $F(C_i) = C_{i+1}$ for some $1 \leq i \leq 2^n$, then because of the feisel structure, one gets $L_{i+1} = R_i$. This means that in order to find a slid pair of (P_i, C_i) one needs to search slid pair only in the set of all pairs which satisfy the condition of $L_{i+1} = R_i$. If this condition is not satisfied, there is no need to check whether they are slid pairs or not. This is the filtering condition of slide attack in Feistel Structures.

3.2.1 Chosen - Plaintext Attack to Feistel Structures

Slide attack can be improved as mentioned at previous section by using chosen-plaintexts. The secret key can be found with probability $P = 1$ upon this improvement.

Fix any (P, C) where P consist of two halves $P = (P_L, P_R)$ and C also consist of two halves $C = (C_L, C_R)$. If (P, C) and (P', C') form a slid pair then, the condition that mentioned at below must be satisfied:

$$\begin{aligned} P_R &= P'_L \\ C_R &= C'_L \end{aligned}$$

Consider $(P^{'}, C^{'})$ where $P_R = P_L^{'}$. This collection has exactly $2^{n/2}$ elements out of all collections. In this collection to identify the unique slid pair $(P^{'}, C^{'})$, one can encrypt whole candidate pairs and check the other filtering condition which is $C_R = C_L^{'}$. Note that this filters one element out of $2^{n/2}$. Thus, it is expected that only one element from this collection will satisfy the condition of being a slid pair. One the other hand, since this collection contains only exactly one slid pair, the success probability of this attack $P = 1$. The attack is done with $2^{n/2}$ encryption and $2^{n/2}$ chosen-plaintexts.

Algorithm 1 Chosen-plaintext attack to Feistel

1: For fixed known (P, C), encrypt $i = 2^{n/2}$ chosen-plaintext where $P_R = P_L^i$
2: Store in the hash tables (C_L^i, P^i, C^i) indexed the by the first coordinate
3: **for** Each chosen plaintext/ciphertext pairs (P^i, C^i) **do**
4: Check the condition $C_R = C_L^{'}$
5: **if** A collision is found **then**
6: Extract the K and check $E_K(P) = C$ or not
7: **end if**
8: **end for**

Another approach to find a slid pair that one can fix $n/2$ arbitrary bits of P_R of P. If $P = (P_L, P_R)$ and $P^{'} = (P_L^{'}, P_R^{'})$ form a slid pair after one round encryption then $P_R = P_L^{'}$ and $P_R^{'} = P_L \oplus f(P_R, K)$ would be satisfied. After this observation, one can choose $2^{n/4}$ - many P^i and P^j where the P_L^i and P_R^j chosen randomly. Hence $2^{n/4} + 2^{n/4} = 2^{n/4+1}$ chosen plaintexts are needed to find a slid pair with high probability by generalized birthday paradox[19].

Furthermore, slid pair would be found by searching the condition ${C^i}_R = {C^j}_L$ between the corresponding ciphertexts. According to naive approach, this searching process that one by one comparison has $2^{n/4} * 2^{n/4} = 2^{n/2}$ time complexity. However, this number can be reduced by some sorting algorithms such as Quick Sort, Merge Sort. For instance, If Quick Sort is used then time complexity of comparison would be reduced to $\mathcal{O}(n^2)$. After these processes, If one found a collusion then he can verify these pairs are slid pair by partial encryption.

CHAPTER 4

APPLICATIONS OF SLIDE ATTACK

4.1 Slide Attack on 2K-DES

The first example where the slide attack applicable is the encryption algorithm called 2K-DES that is the variant of Data Encryption Standard[4]. Following table summarizes the differences of 2K-DES from the original DES. By increasing the number

Table 4.1: Changes of DES to 2K-DES

	DES	2K-DES
Block Size	64 bits	64 bits
Key Size	56 bits	96 bits
Round	16	64
Key Schedule	Generate 48-bits round key for every rounds	In the odd rounds K_1 is used In the even rounds K_2 is used

of rounds, the algorithm probably gain the resistance against to linear and differential attack. Additionally, new key size make the algorithm immune to exhaustive search. However, the usage of partial keys in every odd and even rounds means that there is an iteration between odd or even rounds. Additionally, Feistel Function has a vulnerability to 2 known-plaintexts. These reasons make 2K-DES a suitable algorithm for slide attack.

4.1.1 Attack Scenario

For any known-plaintext (P, C), one can decrypt C to all possible 2^{32} result of C' with K_2. Each C' contains the left 32-bits of C on the right side of itself because of the feistel structure. Note that, DES decryption is the same as encryption routine.So it is appropriate that the attacker can request encrypt all of $E_K(C') = P'$ to obtain corresponding 2^{32} of P's. These P's are equivalent to one round back to $F_{K_2}^{-1}(P) = P'$.

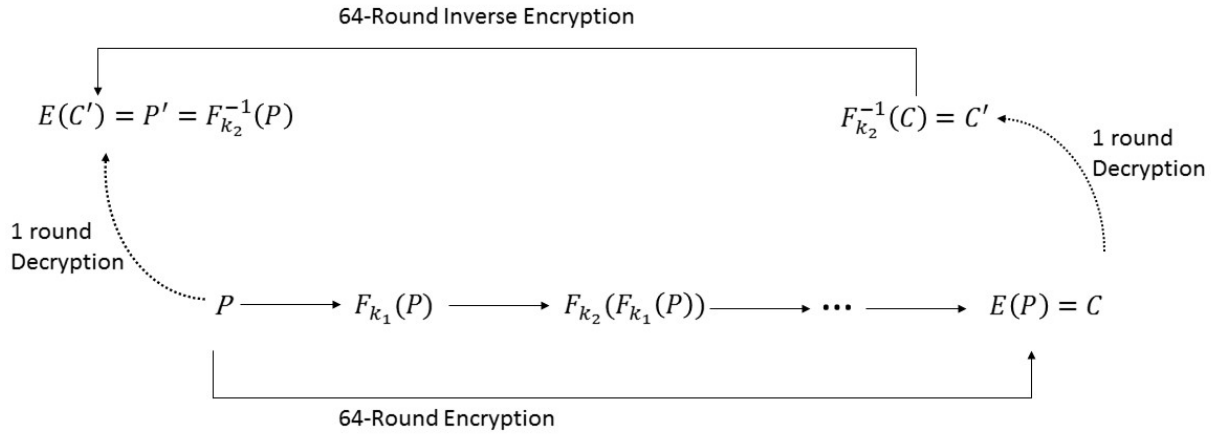

Figure 4.1: Attack Scheme

Since the left 32-bit of the P should be right 32-bit of the P' due to the feistel network, one can check this condition on the 2^{32} pool of P' and we can expect that almost all of wrong guesses of (P', C') pairs must be eliminated. When the right pair of (P', C') is found one can derive the K_2 from the $F_{K_2}(P') = P$ and $F_{K_2}(C') = C$ equations. These operations are made for extract the true K_2 with high probability. Additionally, the attacker can derive the other half of K, K_1 , with exhaustive search. However, this search increases the time complexity extremely. Instead of exhaustive search, one can repeat the attack for find K_1 with the help of known K_2.

Algorithm 2 A Slide Attack on 2K-DES

```
Ask for the encryption of a known plaintext (P, C);
for All C^i where C_R^i = C_L and i = 1...2^32 do
    Encrypt C^i to P^i ;
    if The condition P_R^i = P_L is satisfied then
        Extract the K_2 from F_{K_2}^{-1}(P) = P^i and F_{K_2}^{-1}(C) = C^i ;
    else
        i++ ;
    end if
end for
```

4.1.2 Complexities

During the attack process, we used 2^{32} adaptively chosen plaintext - ciphertext pair and encrypted 2^{32} many C' to P' for finding each K_1 and K_2. Thus total data complexity for finding secret K is $2^{32} + 2^{32} = 2^{33}$ adaptive chosen plaintexts and total time complexity is $2^{32} + 2^{32} = 2^{33}$ DES encryption.

4.2 A Simple Attack on 1K-AES

1K-AES[1] consists of almost same structures with conventional AES except its arbitrary number of rounds. We don't ignore last MixColumn operation at the encryption process with 1K-AES. Additionally, the same key is used in each round of algorithm. It means that there is 1 round self-similarity so that f function consists only 1 round. The symbolization of 1K-AES encryption and f function are as below.

$$E_k(x) = F_k \circ F_k \circ ... \circ F_k(x) \oplus k \text{ where } F_k(x) = MC \circ SR \circ SB \circ ARK_k(x)$$

$$E_k(x) = G_k \circ G_k \circ ... \circ G_k(x \oplus k) \text{ where } G_k(x) = ARK_k \circ MC \circ SR \circ SB(x)$$

4.2.1 Attack Scenario

It is mentioned above that if f_k is weak, then k can be recovered easily with using two input-output pairs. This circumstance for 1K-AES can be shown like that;

If (P, C) and $(P^{'}, C^{'})$ form a slid pair under 1K-AES then;

$$\left.\begin{array}{ll} F_{k_1}(P) = P^{'} \implies k_1 & \text{can be obtained.} \\ G_{k_2}(C) = C^{'} \implies k_2 & \text{can be obtained.} \end{array}\right\} k_1 = k_2$$

Additionally, It would be beneficial that is mentioned about some observations.

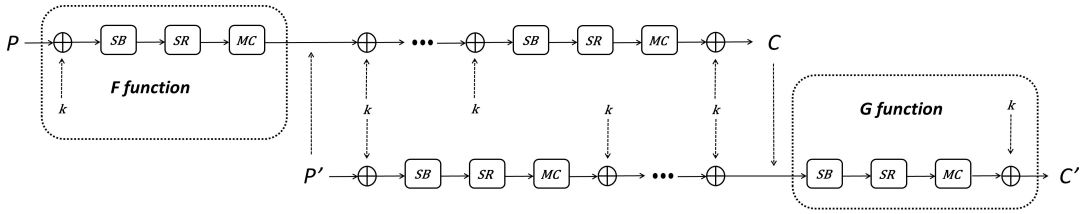

Figure 4.2: Attack Scheme

Assume that (P, C) and $(P^{'}, C^{'})$ is a slid pair where $F_k(P) = P^{'}$ and $G_k(C) = C^{'}$
This implies;
$F_k(P) = MC \circ SR \circ SB \circ ARK_k(P) = MC \circ SR \circ SB(P \oplus k) = P'$ and
$G_k(C) = ARK_k \circ MC \circ SR \circ SB(C) = k \oplus (MC \circ SR \circ SB(C)) = C'$.

Let $F_k(P) = P^{'}$. Then;
$F_k^{-1}(P^{'}) = P = k \oplus (SB^{-1} \circ SR^{-1} \circ MC^{-1}(P^{'}))$
Let $\overline{P'} = SB^{-1} \circ SR^{-1} \circ MC^{-1}(P^{'})$ then;
$P = k \oplus \overline{P'} \implies P \oplus \overline{P'} = k$

Moreover, corresponding ciphertexts C and $C^{'}$ are also satisfy the same property.

Let $G_k(C) = C^{'}$. Then; $G_k(C) = k \oplus (MC \circ SR \circ SB(C)) = C^{'}$
$\implies C^{'} = k \oplus (MC \circ SR \circ SB(C))$

Let $\overline{C} = MC \circ SR \circ SB(C)$ then;
$C' = k \oplus \overline{C} \implies C' \oplus \overline{C} = k$

After that, combining these equations $P \oplus \overline{P'} = k$ and $C' \oplus \overline{C} = k$

$P \oplus \overline{P'} = C' \oplus \overline{C}$
$\implies P \oplus \overline{C} = \overline{P'} \oplus C'$ is found as a filtering condition.

With this informations, one can find some equations that are useful in order to find a filtering condition between the Plaintext-Ciphertext collections which will be used during the attack process.

This filtering condition will be used in order to eliminate the wrong estimations for being a slid pair. The attack algorithm is at Figure 4.2.

Algorithm 3 A Slide Attack on 1K-AES

1: Ask for the encryption of 2^{64} known plaintexts (P_i, C_i)
2: **for** Each plaintext/ciphertext pairs (P_i, C_i) **do**
3: Compute the value $\overline{C_i} = MC \circ SR \circ SB(C_i)$
4: Compute the value $P_i \oplus \overline{C_i}$
5: Store in the hash table the pairs $(P_i \oplus \overline{C_i}, P_i)$
6: **end for**
7: **for** Each plaintext/ciphertext pairs (P_j, C_j) **do**
8: Compute the value $\overline{P_j} = SB^{-1} \circ SR^{-1} \circ MC^{-1}(P_j)$
9: Compute the value $\overline{P_j} \oplus C_j$
10: Check for entries in the hash table whose first coordinate matches it
11: **end for**
12: **for** Each collision in the table $(P_i \oplus \overline{C_i} = \overline{P_j} \oplus C_j)$ **do**
13: Check the candidate key $K = P_i \oplus \overline{P_j}$
14: **end for**

4.2.2 Complexities

We expect that these 2^{64} known plaintexts contain a slid pair with high probability by birthday paradox. Total data, time and memory complexities are both 2^{64}.

4.3 Slide Attack on Reduced-PRESENT

Reduced-PRESENT which we defined, has almost same structures with conventional PRESENT. The only difference between them is that Reduced-PRESENT is performed with same 64-bit round-key in every round. Note that, number of rounds is unimportant in our attack so that Reduced-PRESENT doesn't perform with a specific number of rounds.

4.3.1 Attack Scenario

Reduced-PRESENT is a lightweight block cipher based on Substitution-Permutation Network. As we considered in Chapter 3, one should determine the filtering condition between the slid pairs.

Definition 4.1. Let E_k be an encryption of Reduced-PRESENT. In this sense, $r - round$ encryption $E_k = F_k^r(P \oplus k) = C$ where $F_k(x) = Per(Sub(x)) \oplus k$, Sub is the substitution layer, Per is the permutation layer and k is the round-key of Reduced-PRESENT.

As it can be seen in the Figure 3.2, a slid pair satisfies the conditions that are shown below;

Let (P, C) and $(P^{'}, C^{'})$ forms a slid pair. Then,

$$\begin{aligned} &F_k(P \oplus k) = P^{'} \oplus k \implies Per(Sub(P \oplus k)) \oplus k = P^{'} \oplus k \\ &\implies Per(Sub(P \oplus k)) = P^{'} \implies P = Sub(Per(P^{'})) \oplus k \\ &\text{Denoting } Sub(Per(P^{'})) = \overline{P^{'}}\text{,we have} \\ &P = \overline{P^{'}} \oplus k \implies P \oplus \overline{P^{'}} = k \end{aligned} \tag{4.1}$$

Moreover;

$$\begin{aligned} &F_k(C) = C^{'} \implies Per(Sub(C)) \oplus k = C^{'} \\ &\text{Denoting } Per(Sub(C)) = \overline{C}\text{,we have} \\ &\overline{C} \oplus k = C^{'} \implies \overline{C} \oplus C^{'} = k \end{aligned} \tag{4.2}$$

After combining these two equations, we obtain a filtering condition:

$$P \oplus \overline{P^{'}} = k = \overline{C} \oplus C^{'} \implies P \oplus \overline{C} = \overline{P^{'}} \oplus C^{'} \tag{4.3}$$

The filtering condition shows that, if an attacker calculate right-side and left-side of the equation for 2^{32}(by birthday paradox) known plaintext-ciphertext pairs, she/he can check which pairs form a slid pair. The attack algorithm is given as below.

Algorithm 4 Slide Attack on Reduced-PRESENT

1: Ask for the encryption of 2^{32} known plaintexts (P_i, C_i)
2: **for** Each plaintext/ciphertext pairs (P_i, C_i) **do**
3: Compute the value $\overline{C_i} = Per(Sub(C_i))$
4: Compute the value $P_i \oplus \overline{C_i}$
5: Store in the hash table the pairs $(P_i \oplus \overline{C_i}, P_i)$
6: **end for**
7: **for** Each plaintext/ciphertext pairs (P_j, C_j) **do**
8: Compute the value $\overline{P_j} = Sub^{-1}(Per^{-1}(P_j))$
9: Compute the value $\overline{P_j} \oplus C_j$
10: Check for entries in the hash table whose first coordinate matches it
11: **end for**
12: **for** Each collision in the table $(P_i \oplus \overline{C_i} = \overline{P_j} \oplus C_j)$ **do**
13: Check the key candidate $k = P_i \oplus \overline{P_j}$ with partial encryption
14: **end for**

4.3.2 Complexities

In this attack, one needs 2^{32} known plaintext-ciphertext pairs by birthday paradox. Additionally, there are 2^{32} entry in the hash table, which are used for find a collusion. There are 2^{32} operations for mount this attack.

4.4 Slide Attack on KeeLoq

The attack of Dunkelman and others on KeeLoq[13] combines slide attack with meet in the middle approach. The aim is to recover key bits from slid pairs. Recall that slide attack depends on two assumptions. The first one is weak key schedule and the other one is F function which is weak to two known plaintext-ciphertext pairs.

As stated before, in the KeeLoq encryption, there are 528 rounds and in each round one bit of 64-bit key is used. This means that every 64 rounds, the same key is replicated. This is a major weakness in key schedule. In this approach, F function can be taken as 64 rounds of KeeLoq. However, 528 is not a multiple of 64.

This problem can be solved by guessing the 16 least significant bits of the key. Since the key is repeated every 64 rounds, in the last 16 rounds, the first 16 bits of the key is reutilized. In this way, the last 16 rounds can be ignored and the cipher can be thought as 512 rounds. Thus, the attack can be mounted with 512 rounds of KeeLoq. From now on, the factor of 2^{16} is added to the complexity of the attack for the key guess.

The block size of KeeLoq is 32 bits. By Birthday Paradox, in a random collection of $\sqrt{2^{32}} = 2^{16}$ plaintext-ciphertext pairs, there exists at least one slid pair with high probability. Thus, 2^{16} plaintext-ciphertext pairs are needed to perform the attack. Note that it is not a time-consuming process to obtain that amount of pairs if appropriate tools are used.

4.4.1 Determining the Key Bits

Before mounting the attack, Linear Step of KeeLoq[7], described by Bogdanov, should be explained. According to his description, all key bits of 32 rounds (or less) can be recovered provided that the two intermediate states which are separated by 32 rounds (or less) are known in the encryption. To understand this approach, the encryption process is examined.

Let $Y^{(i)} = (y_{31}^{(i)}, .., y_0^{(i)})$ and $Y^{(i+t)} = Y^{(i)} = (y_{31}^{(i+t)}, .., y_0^{(i+t)})$ be the two known states, $t \leq 32$. After one round encryption, the result is

$$Y^{(i+1)} = (y_{31}^{(i+1)}, y_{30}^{(i+1)}, y_{29}^{(i+1)}, .., y_0^{(i+1)}) = (\varphi^{(i)}, y_{31}^{(i)}, y_{30}^{(i)}, y_{29}^{(i)}, ..., y_1^{(i)}) \tag{4.4}$$

$$\varphi^{(i)} = NLF(y_{31}^{(i)}, y_{26}^{(i)}, y_{20}^{(i)}, y_9^{(i)}, y_1^{(i)}) \oplus y_{16}^{(i)} \oplus y_0^{(i)} \oplus k_{i(mod 64)} \tag{4.5}$$

Because of the $NLFSR$ structure of the round function and since $0 < t \leq 32$, the bit $\varphi^{(i)} = y_{32-t}^{(i+t)}$. This encrypted value is one of the bits of $Y^{(i)}$ and thus known. Hence,

$$k_{i(mod64)} = NLF(y_{31}^{(i)}, y_{26}^{(i)}, y_{20}^{(i)}, y_{9}^{(i)}, y_{1}^{(i)}) \oplus y_{16}^{(i)} \oplus y_{0}^{(i)} \oplus y_{32-t}^{(i+t)} \tag{4.6}$$

This result is satisfied by the XOR operation. In KeeLoq, if the key bits were used in NLF or some kind of a non-linear structure, this method should not work. By repeating this t times, we can recover t consecutive bits of the key. For instance, for $t = 3$ by assuming $Y^{(3)}$ = ($y_{31}^{(3)}$,..., $y_{0}^{(3)}$) is known, we can recover 3-bits of key as follows.

With the three times encryption routine we obtain:

$$\varphi^{(0)} = y_{29}^{(3)}, \varphi^{(1)} = y_{30}^{(3)}, \varphi^{(2)} = y_{31}^{(3)} \tag{4.7}$$

It means that;

$$Y^{(3)} = (y_{31}^{(3)}, y_{30}^{(3)}, y_{29}^{(3)}, .., y_{0}^{(3)}) = (\varphi^{(2)}, \varphi^{(1)}, \varphi^{(0)}, y_{31}^{(0)}, ..., y_{3}^{(0)}) \tag{4.8}$$

As a result,

$$\varphi^{(0)} = y_{29}^{(3)} = NLF(y_{31}^{(0)}, y_{26}^{(0)}, y_{20}^{(0)}, y_{9}^{(0)}, y_{1}^{(0)}) \oplus y_{16}^{(0)} \oplus y_{0}^{(0)} \oplus k_0 \Rightarrow \tag{4.9}$$

$$k_0 = NLF(y_{31}^{(0)}, y_{26}^{(0)}, y_{20}^{(0)}, y_{9}^{(0)}, y_{1}^{(0)}) \oplus y_{16}^{(0)} \oplus y_{0}^{(0)} \oplus y_{29}^{(3)} \tag{4.10}$$

And similarly, we obtain;

$$\varphi^{(1)} = y_{30}^{(3)} = NLF(y_{31}^{(1)}, y_{26}^{(1)}, y_{20}^{(1)}, y_{9}^{(1)}, y_{1}^{(1)}) \oplus y_{16}^{(1)} \oplus y_{0}^{(1)} \oplus k_1 \Rightarrow \tag{4.11}$$

$$k_1 = NLF(y_{31}^{(1)}, y_{26}^{(1)}, y_{20}^{(1)}, y_{9}^{(1)}, y_{1}^{(1)}) \oplus y_{16}^{(1)} \oplus y_{0}^{(1)} \oplus y_{30}^{(3)} \tag{4.12}$$

Moreover;

$$\varphi^{(2)} = y_{31}^{(3)} = NLF(y_{31}^{(2)}, y_{26}^{(2)}, y_{20}^{(2)}, y_{9}^{(2)}, y_{1}^{(2)}) \oplus y_{16}^{(2)} \oplus y_{0}^{(2)} \oplus k_2 \Rightarrow \tag{4.13}$$

$$k_2 = NLF(y_{31}^{(2)}, y_{26}^{(2)}, y_{20}^{(2)}, y_{9}^{(2)}, y_{1}^{(2)}) \oplus y_{16}^{(2)} \oplus y_{0}^{(2)} \oplus y_{31}^{(3)} \tag{4.14}$$

Hence, we obtain 3-bits of the key. While a slid pair is being searched in the F permutation during a random attack, the process will be divided into 16-round parts. For this reason, this figure will be useful.

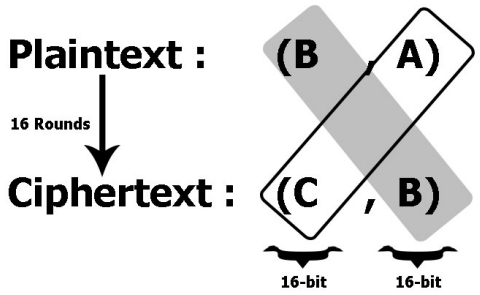

Figure 4.3: 16 rounds of KeeLoq encryption

Shown at the Figure 4.3, 16-round result of plaintext(denoted by ciphertext) shows that 16 most significant bits (MSB) of plaintext (denoted by B) will be equal to 16-round LSB of ciphertext due to the KeeLoq encryption routine at the end. In addition, as mentioned above, if an attacker has knowledge about A and C then 16-bit of the key can be obtained.

4.4.2 The Attack Scenario

In this section, the basic attack scenario of KeeLoq is explained.

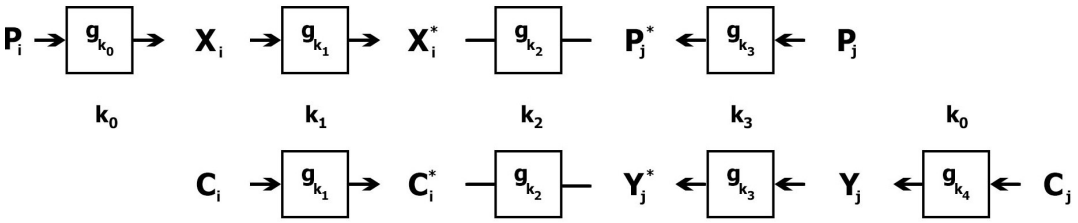

Figure 4.4: Notation of the attack

As stated above, 2^{16}-many plaintext-ciphertext pairs are needed for the attack by Birthday Paradox. For the definition of F function, 64 rounds of KeeLoq are taken. Moreover, F is divided into four parts as in the Figure 4.4.Thus, the attacker can seek an equality P_j^*(16-LSB of the P_j^*) = $\overline{X_i^*}$(16-MSB of the X_i^*) as a condition for being a slid pair. Therefore, the 64-bit key k is split into four equal parts such that $k =< k_3, k_2, k_1, k_0 >$, where k_0 contains 16 least significant key bits.

Note that if $F(C_i) = C_j$ when $F(P_i) = P_j$ for the two plaintext-ciphertext pairs (P_i, C_i) and (P_j, C_j), they form a slid pair. Since there are 2^{16} plaintext-ciphertext pairs, there are 2^{16} comparisons of being a slid pair for each pair. Therefore, there are $2^{16}.2^{16} = 2^{32}$ match-ups to be checked. This shows that the probability of finding a true match-up, i.e. a slid pair is $1/2^{32}$. Thus, the probability of not finding a true match-up is $1 - 1/2^{32}$. In the set of all possible slid pairs, there are $2^{32}.2^{32} = 2^{64}$ elements and from these elements, 2^{32} pairs are taken to look for a slid pair. Therefore, there are 2^{32} sets to look for an existence of a slid pair. In the worst case scenario, a slid pair may not found until the last set. This gives the probability of not finding a slid pair all over 2^{32} sets as $(1 - 2^{-32})^{2^{32}}$. Hence, the probability of finding a slid pair all over these sets is $1 - (1 - 2^{-32})^{2^{32}} \approx 0.63$ or in other words, the success probability of the attack is $\%63$.

Algorithm 5 Slide Attack on KeeLoq

for all $k_0 \in \{0,1\}^{16}$ **do**
  for all Plaintexts P_i, $0 \leq i < 2^{16}$ **do**
    Partially encrypt P_i to X_i
    Partially decrypt C_i to Y_i
  end for
  for all $\overline{P_j^*} \in \{0,1\}^{16}$ **do**
    for all Plaintexts P_j, $0 \leq j < 2^{16}$ **do**
      Determine the key bits k_3
      Partially decrypt Y_j to Y_j^*
      Save the tuple $< P_j^*, C_j^*, k_3 >$ in a table
    end for
    for all Plaintexts P_i, $0 \leq i < 2^{16}$ **do**
      Determine the key bits k_1
      Partially encrypt C_i to C_i^*
      for all Collisions $\overline{C_i^*} = \overline{Y_j^*}$ in the table **do**
        Determine the key bits k_2 from X_i^* and P_j^*
        Determine the key bits $k_2^{'}$ from C_i^* and Y_j^*
        if $k_2 = k_2^{'}$ **then**
          Encrypt 2 known plaintexts with the key $k =< k_3, k_2, k_1, k_0 >$
          if the correct ciphertexts are found **then**
            return The key is k
          end if
        end if
      end for
    end for
  end for
end for
return Failure

In this part, it will be more convenient to explain the attack step by step. Note that, there are some illustrations that related to every step of the attack in Appendix part.

1. First, as explained above, one should start the process by guessing the partial key k_0. From a set of 2^{16} key options, the attacker chooses one k_0. After that, all of 2^{16} many P_i's are partially encrypted to X_i's. Similarly, 2^{16} many C_j's are partially decrypted to Y_j's with the same k_0. In short, the whole attack starts with the prediction of k_0 and by k_0, a slid pair is searched throughout the process. In the case of not finding a slid pair or finding the wrong one, it is necessary to try the other possibilities of k_0 until the correct key is found. This means that, one should have to try all 2^{16} many k_0's in the worst case scenario.

2. Secondly, applying the meet-in-the-middle approach to the attack will be suitable at this stage. In this step, one should guess the LSB of P_j^* ($\overline{P_j^*}$) from the all 2^{16} many options so that 2^{16} many k_3 can be determined for 2^{16} many P_j's. As it is

mentioned in Figure 4.3, if P_j^* and P_j are known, the attacker can obtain the 16-bit key used in the encryption. Having all 2^{16} many P_j's and knowing $\overline{P_j^*} = \underline{P_j}$, the only remained thing is to guess the unknown $\underline{P_j^*}$. Then, for all possible P_j^*'s, the corresponding 2^{16} many k_3's are determined. Consequently a table which includes the values of P_j^* with unique k_3 is created. Afterwards, by using k_3's, all Y_j's are decrypted to Y_j^*'s. The values of Y_j^* are also added to this table. Hence, one has the tuple $< P_j^*, C_j^*, k_3 >$ for every plaintext P_j. This table will be used to catch a collisions between the other parts of the key that used both in encryption and decryption.

3. In the third step, something similar to the second step is done from the other side. Since all bits of X_i and $\underline{X_i^*}$ are known and since the equality $\overline{X_i^*} = \underline{P_j^*}$ should be satisfied for all $\underline{P_j^*}$, the key bits of k_1 can be determined with the formulas used in Section 4.4.1.

   To be more clear, for the determined 2^{16} many X_i's (these are determined in step one with the guess of k_0), there is one option for the equality $\overline{X_i^*} = \underline{P_j^*}$. Therefore, there are uniquely determined 2^{16} many k_1's corresponding to each encryption for each plaintext. Hence, to search the second equality for being a slid pair, C_i's are partially encrypted to C_i^*'s with each k_1 and the results of encryption are also recorded to the table constructed in previous step.

   Note that if P_i and P_j forms a slid pair, their partial encryptions and decryptions (under the correct key k) must meet in the middle. It means that if a pair is a slid pair, then it should satisfy $\overline{X_i^*} = \underline{P_j^*}$ and also $\overline{C_i^*} = \underline{Y_J^*}$. In this sense, k_1 satisfies the first equality. To check for the second equality, one should compare the result of encryption $\overline{C_i^*}$ and $\underline{Y_j^*}$ to catch a collision in the table. If a collision is caught, the corresponding pairs (P_i, C_i) and (P_j, C_j) become a candidate slid pair. This circumstance can be observed only if the predictions for k_0 and $\underline{P_j^*}$ are true.

4. To sum up, one should check the candidate slid pairs against false alarm. With the candidate slid pairs, one should determine k_2 which is used between X_i^* and P_j^* and $k_2^{'}$ which is used between C_i^* and Y_j^*. Knowing all X_i^* , P_j^* , C_i^* and Y_j^*,one can determine these k_2 and $k_2^{'}$. If the equality of k_2 = $k_2^{'}$ is caught, this pair will be a slid pair otherwise, it will not. Hence, with the determined k = $< k_3, k_2, k_1, k_0 >$, the condition of $F_k(P_i) = P_j$ and $F_k(C_i) = C_j$ is satisfied.

5. The secret key k may not be the right key even if it satisfies the condition of being a slid pair. In order to check this situation, a plaintext-ciphertext pair which is chosen from set of all 2^{16}-many known pairs, can be used. If the encryption of plaintext gives the correct ciphertext, then the secret key k can be true, otherwise it means a false alarm.

4.4.3 Complexity

This study shows that the protocol "Identify Friend or Foe"- based KeeLoq is not secure with 2^{16}-known plaintexts and 2^{45}-KeeLoq encryptions. The implementation of the attack is fully running about 500 days for known-plaintexts with a CPU core. It means that with x-CPU cores, the attack runs 500/x days for known-plaintexts and 218/x days for chosen-plaintexts. For example, if an attacker uses 50 CPU cores in parallel, KeeLoq can be broken in 500/50 = 10 days.

Complexity
Data Complexity: 2^{16} known plaintexts
Memory: $\pm$ 2 MB for the 2^{16}-tuples $< P_j^*, Y_j^*, k_3 >$
Time: 2^{45} KeeLoq encryptions

CHAPTER 5

CONCLUSION

Undoubtedly, slide attack is an important method for the cryptanalysis of block ciphers. It uses weaknesses of key schedules and it is independent of the number of rounds. So, these reasons also make slide attack effective for lightweight block ciphers which have high number of rounds against differential and linear cryptanalysis.

In this thesis, mathematical fundamentals of slide attack are described in detail. It is also shown that slide attack is more effective if it is combined with some approaches such as finding a filtering condition, meet-in-the-middle or Birthday Paradox. Besides, some special cases of slide attack on Substitution-Permutation Network's or Feistel Structures are investigated. Moreover, some applications of slide attack to reduced block ciphers are given. Additionally, we offer a practical attack to reduced-PRESENT inspired from. Lastly, a real lightweight block cipher KeeLoq and how it is broken with slide attack is explained.

REFERENCES

[1] A. Bar-On, E. Biham, O. Dunkelman, and N. Keller, Efficient slide attacks, Journal of Cryptology, Aug 2017.

[2] E. Biham and A. Shamir, Differential cryptanalysis of des-like cryptosystems, in *Advances in Cryptology-CRYPTO*, volume 90, pp. 2–21, Springer, 1991.

[3] E. Biham and A. Shamir, *Differential cryptanalysis of the data encryption standard*, Springer Science & Business Media, 2012.

[4] A. Biryukov and D. Wagner, Slide attacks, in *International Workshop on Fast Software Encryption*, pp. 245–259, Springer, 1999.

[5] A. Biryukov and D. Wagner, Advanced slide attacks, in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 589–606, Springer, 2000.

[6] A. Bogdanov, Attacks on the keeloq block cipher and authentication systems, in *3rd Conference on RFID Security*, volume 2007, 2007.

[7] A. Bogdanov, Cryptanalysis of the keeloq block cipher., IACR Cryptology ePrint Archive, 2007, p. 55, 2007.

[8] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe, Present: An ultra-lightweight block cipher, in *CHES*, volume 4727, pp. 450–466, Springer, 2007.

[9] J. Daemen and V. Rijmen, The block cipher rijndael, in *CARDIS*, volume 1820, pp. 277–284, Springer, 1998.

[10] M. H. K. C. H. Encoder and D. D. Sheet, Microchip technology inc, 2002.

[11] H. Feistel, Cryptography and computer privacy, Scientific american, 228(5), pp. 15–23, 1973.

[12] H. Feistel, Block cipher cryptographic system, March 19 1974, uS Patent 3,798,359.

[13] S. Indesteege, N. Keller, O. Dunkelman, E. Biham, and B. Preneel, A practical attack on keeloq, Advances in Cryptology–EUROCRYPT 2008, pp. 1–18, 2008.

[14] J. B. Kam and G. I. Davida, Structured design of substitution-permutation encryption networks, IEEE Transactions on Computers, (10), pp. 747–753, 1979.

[15] M. Matsui, Linear cryptanalysis method for des cipher, in *Workshop on the Theory and Application of of Cryptographic Techniques*, pp. 386–397, Springer, 1993.

[16] F. Mosteller, Understanding the birthday problem, in *Selected Papers of Frederick Mosteller*, pp. 349–353, Springer, 2006.

[17] N. F. PUB, 46-3. data encryption standard, Federal Information Processing Standards, National Bureau of Standards, US Department of Commerce, 1977.

[18] V. Rijmen and J. Daemen, Advanced encryption standard, Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology, pp. 19–22, 2001.

[19] D. Wagner, A generalized birthday problem, in *Crypto*, volume 2442, pp. 288–303, Springer, 2002.

APPENDIX A

Figures of Slide Attack to KeeLoq

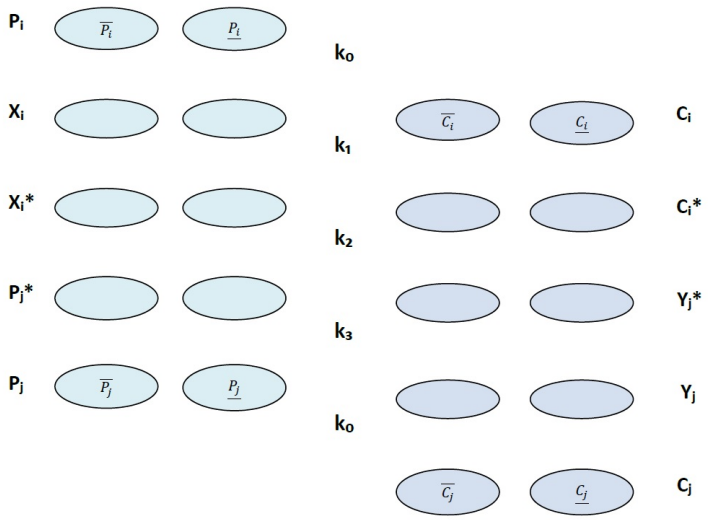

Figure A.1: Beginning of the attack

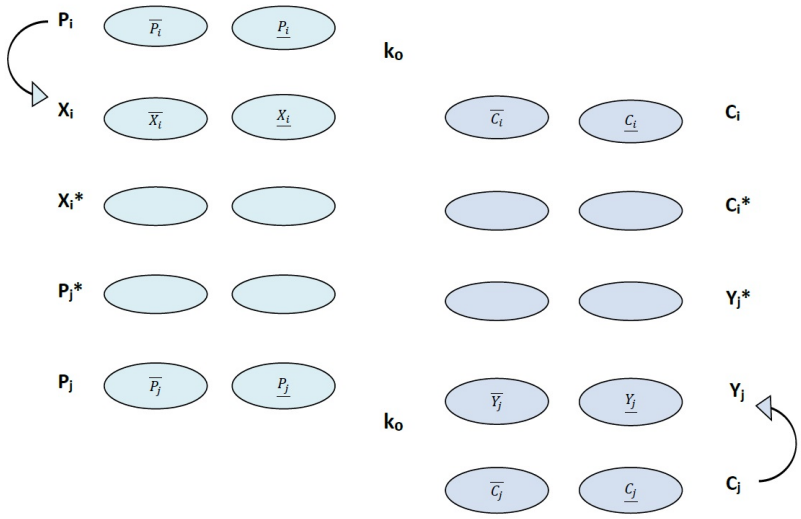

Figure A.2: Illustration of Step 1

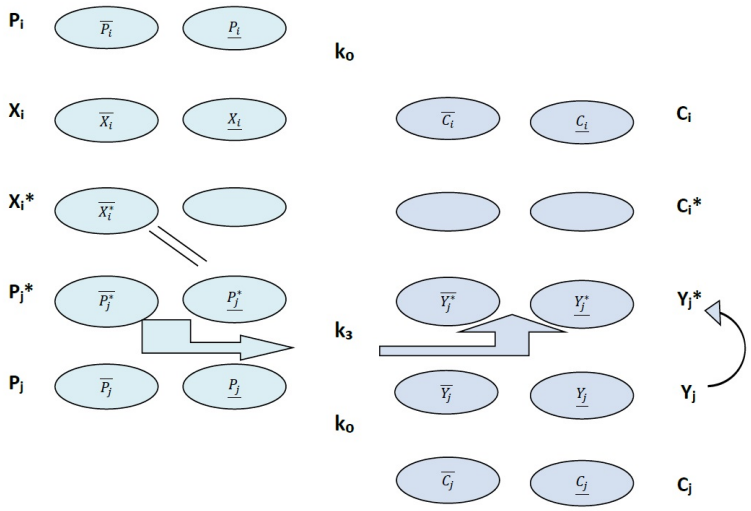

Figure A.3: Illustration of Step 2

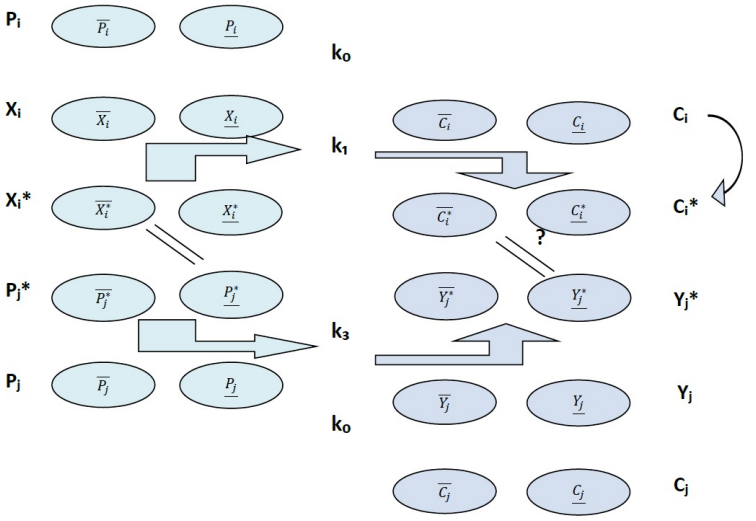

Figure A.4: Illustration of Step 3

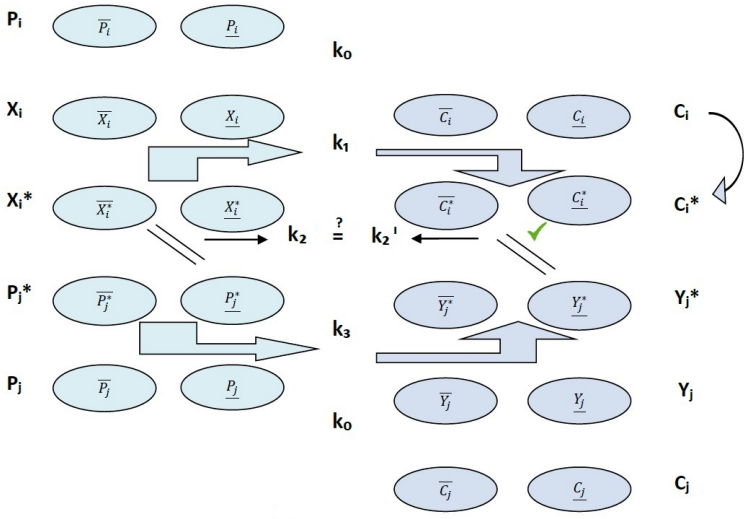

Figure A.5: Illustration of Step 4