ON THE SECURITY OF TIGER HASH FUNCTION

ONUR ÖZEN

JANUARY 2008

ON THE SECURITY OF TIGER HASH FUNCTION

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

BY

ONUR ÖZEN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF CRYPTOGRAPHY

JANUARY 2008

Approval of the Graduate School of Applied Mathematics

Prof. Dr. Ersan AKYILDIZ
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Ferruh Özbudak
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Ali DOĞANAKSOY
Supervisor

Examining Committee Members

Prof. Dr. Ersan AKYILDIZ

Prof. Dr. Ferruh ÖZBUDAK

Assoc. Prof. Dr. Ali DOĞANAKSOY

Assis. Prof. Dr. Ali Aydın SELÇUK

Dr. Muhiddin UĞUZ

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 : Onur ÖZEN
Signature :

Abstract

ON THE SECURITY OF TIGER HASH FUNCTION

Özen, Onur
M.Sc., Department of Cryptography
Supervisor: Assoc. Prof. Dr. Ali DOĞANAKSOY

January 2008, 86 pages

Recent years have witnessed several real threats to the most widely used hash functions which are generally inspired from `MD4`, such as `MD5, RIPEMD, SHA0` and `SHA1`. These extraordinary developments in cryptanalysis of hash functions brought the attention of the cryptology researchers to the alternative designs. `Tiger` is an important type of alternative hash functions and is proved to be secure so far as there is no known collision attack on the full (24 rounds) `Tiger`. It is designed by Biham and Anderson in 1995 to be very fast on modern computers.

In two years some weaknesses have been found for `Tiger`-hash function. First, in FSE ′06 Kelsey and Lucks found a collision for 16-17 rounds of `Tiger` and a pseudo-near-collision for 20 rounds. Then, Mendel *et al* extended this attack to find 19-round collision and 22-round pseudo-near-collision. Finally in 2007, Mendel and Rijmen found a pseudo-near-collision for the full `Tiger`. In this work, we modify the attack of Kelsey and Lucks slightly and present the exact values of the differences used in the attack.

Moreover, there have been several cryptanalysis papers investigating the randomness properties of the designed hash functions under the encryption modes. In these papers, related-key boomerang and related-key rectangle attacks are performed on MD4,MD5, HAVAL and SHA. In this thesis, we introduce our 17,19 and 21-round related-key boomerang and rectangle distinguishers to the encryption mode of Tiger.

Keywords: Tiger, Cryptanalysis, Hash Functions, Collision, Boomerang Attack

Öz

TIGER ÖZET FONKSİYONUN GÜVENLİĞİ ÜZERİNE

Özen, Onur
Yüksek Lisans, Kriptografi Bölümü
Tez Yöneticisi: Doç. Dr. Ali DOĞANAKSOY

Ocak 2008, 86 sayfa

Son yıllar özellikle `MD4` özet fonksiyon ailesinden türetilmiş `MD5, RIPEMD, SHA0` ve `SHA1` gibi çok kullanılan özet fonksiyonlarına yapılan ataklara tanıklık etmiştir. Bu tip çok kullanılan özet fonksiyonlarına yapılan ataklar kriptoloji araştırmacılarını diğer özet fonksiyonlarına yöneltmiştir. `Tiger`, `MD4` özet fonksiyonu kurgusuna benzer, şu ana kadar güvenli varsayılan 24 çevrimli önemli bir fonksiyondur. `Tiger`, 1995'te Biham ve Anderson tarafından özellikle modern bilgisayarlarda hızlı olacak şekilde tasarlanmıştır.

Son iki yılda `Tiger`' yapısında bazı zayıflıklar bulunmuştur. Öncelikle, FSE '06'da Kelsey ve Lucks `Tiger` özet fonksiyonun 16-17 çevrimi için çakışma ve 20 çevrimi için de sözde-çakışma bulmuştur. Daha sonra bu atak Mendel vd tarafından INDOCRYPT '06'da 19 çevrimlik çakışma ve 22 çevrimlik sözde yarı-çakışmaya geliştirilmiştir. Son olarak da ASIACRYPT '07'de Mendel ve Rijmen 24 çevrimli `Tiger` için sözde-yarı-çakışma bulmuştur. Bu çalışmada, Kelsey ve Lucks'un atağında kullanılan gerçek farklar bulunup, bu atak geliştirilmiştir.

Ayrıca, literatürde özet fonksiyonlarının şifreleme sürümlerinin rassallık özelliklerini inceleyen çalışmalar bulunmaktadır. Bu çalışmalarda, `MD4,MD5`, `HAVAL` ve `SHA` özet fonksiyonlarının şifreleme sürümlerine ilişik anahtarlı bumerang ve dikdörtgen atakları uygulanmıştır. Bu çalışmada, `Tiger`'in şifreleme sürümüne 17, 19 ve 21 çevrimlik ilişik anahtarlı bumerang ve dikdörtgen atakları uygulanmıştır.

Anahtar Kelimeler: Tiger, Kriptanaliz, Özet Fonksiyonları, Çakışma, Bumerang Atağı.

To Nihal and my family,

Acknowledgments

I would like to express my sincere gratitude to my supervisor Ali DOĞANAKSOY for guiding and supporting me like a father, sharing and talking with me like a friend and teaching me life like a mathematician. His way of life has always amazed and excited me.

It is a great pleasure to thank my collegue Kerem VARICI for being with me everytime and everywhere for seven years. Our discussions and sleepless nights gave birth to the valuable contributions to our lives.

I also want to thank Meltem SÖNMEZ TURAN for her amazing ideas and encouregement, Çağdaş ÇALIK for being with me with his valuable skills and his friendship, Çelebi KOCAİR for his collobaration and patience and Fatih SULAK for his partnership.

Very special thanks to Nihal for her love, support and encouregement. I am also grateful to my family for trusting me since I was born and growing up me with their love. I want to thank my friends, everyone at IAM and other members of my family for believing in me.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cryptology is known to be the science of concealing information inspired from Greek words *kryptós* "hidden," and the verb *gráfo* "write" or *legein* "to speak" . It has two basic building blocks, namely the *cryptography* which is the science of designing secure components and the *cryptanalysis* which is science of analyzing or breaking the designed cryptographic primitives. Over the centuries, the history has been witnessed the challenge between these two sciences.

Cryptography has been used over 4000 years to conceal secret and the sensitive information. Throughout the ages, the importance and the way of usage of cryptography has evolved. Before 1960s, when the computers were not the key primitives of our lives, cryptography had been used generally for military purposes. Cryptanalysis, on the other hand, has evolved parallelly with the developments in cryptography. The role of cryptanalysis during the World War I and II affects the future of several countries and the end of these wars was generally dependent to the developments in cryptanalysis.

After 1960s, when the computers has commenced to be vital for communities, the cryptology has started to become public. That is, private sector needed to conceal sensitive information digitally. By these purposes, `DES` (Data Encryption Standard) was designed by IBM as U.S. Federal Information Processing Standard (FIPS) for encrypting unclassified information. It is inspired by the work of Feistel in `Lucifer` and has been used widely since then. In early 2000s, the fast evolution of computers and the recent developments in cryptanalysis lead to a new standard called `AES` (Advanced Encryption Standard) which was designed by Rijmen and Daemon in 1998 and nowadays it is the most widely used publicly known algorithm in the world.

One of the common features of these algorithms is that there exists only one secret parameter in these components, namely the *key*. This secret key is known for each parties who are trying to communicate and used for *encryption* and *decryption*. This type of cryptographic primitive is defined to be *symmetric key primitives*.

In 1976, Diffie and Hellman announced the birth of a new concept called *public key cryptography* which is believed to be the most exciting discovery in cryptology introducing a new concept called *assymmetric key primitives*. They could not exemplify the concept but the idea was interesting. In this smart technique, the uniqueness of the secret key is discarded and distributed over all parties. Namely, every user has decided a secret and a public parameter and the security of the system depends heavily on solving very hard mathematical problems such as discrete logarithm problem.

Rivest, Shamir and Adleman gave the first example of the public-key primitive called `RSA` in 1978 which is based on the difficulty of factoring large integers. After that, El Gamal proposed a new protocol based on discrete logarithm problem. However, the most interesting development came to the picture when digital signatures arise in 1991. Nowadays, the protocols based on public key primitives are one of the hot research topics in cryptology community. The general overview of the cryptographic primitives are given in Figure 1.1 which is taken from the well known reference book [1]: Handbook of Applied Cryptography by A.Menezes, P.van Oorschot and S.Vanstone.

Nowadays, cryptology plays a crucial role in our lives. It became a vital tool for personal information security besides the national security. We use cryptology to protect our sensitive and secret data in many applications such as financial transactions, satellite communications, authentication, e-government applications, etc. In the following years, the role of cryptology is going to become more important and vital as the use of cryptology becomes wide.

The following sections cover the basic building blocks used in this work. In Section 1.1, the block ciphers are summarized and some basics of cryptanalysis methods are given. Hash functions together with their attack scenarios are recapitulated in Section 1.2 and Section 1.3 details our contributions and the structure of the thesis.

1.1 Block Ciphers

A *block cipher* is a symmetric encryption primitive which encrypts n-bit block that is broken from an m-bit message by using k-bit secret key and transmitting n-bit

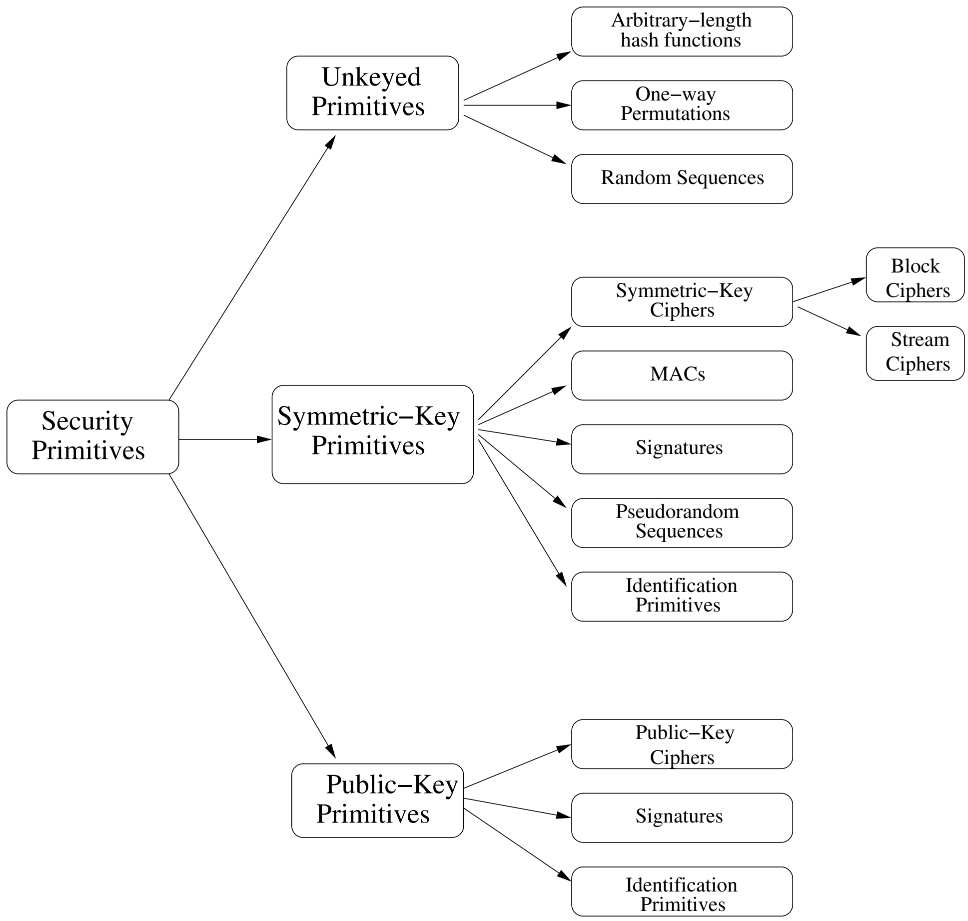


Figure 1.1: Cryptographic Primitives

encrypted data at a time. Most of the encryption primitives used today are block ciphers. Generally, a block cipher is said to encrypt a plaintext of n bits into n-bit ciphertext by using k-bit secret key.

Feistel Ciphers and *Substitution-Permutation-Networks* (SPNs) are two types of block ciphers known today. `DES` is an example of a Feistel Cipher and `AES` can be given as an example of an SPN. Block cipher theory is a very well studied research area and there exist a lot of papers investigating the security of block ciphers. Moreover, cryptanalysis methods special to block ciphers are wide and well established. Throughout this thesis, some of the powerful attack scenarios are going to be presented.

Firstly, some cryptanalysis preliminaries will be given but before that some assumptions need to be done. First of all, an attacker is assumed to know the entire properties of the attacked cipher and the only secret parameter is assumed to be the secret key. This principle is known to be the *Kerkhoff's Principle*. Although this leads to some misunderstandings about the security of the system, this assumption is crucial from the cryptanalysis point of view (in the literature some reverse engineering examples exist, however that is not the case for our model). Below some basic cryptanalytic definitions are presented .

- **Ciphertext-Only Attacks**: In ciphertext only attacks, the attacker is assumed to have just the ciphertexts and there exist minimal knowledge about the plaintext. It is the most powerful attack scenario and nowadays there is no ciphertext-only attacks to block ciphers.

- **Known-Plaintext Attacks**: In this attack scenario, the attacker has access to plaintexts and corresponding ciphertexts. Generally, it is assumed that the attacker gathers some plaintext-ciphertext pairs and tries to reveal the secret key from that knowledge. Linear Cryptanalysis can be given as an example to known-plaintext attacks.

- **Chosen Plaintext Attacks**: In chosen plaintext attacks, the attacker has control over the encryption box. Namely, he is able to choose plaintext and construct the corresponding ciphertexts. Differential Cryptanalysis is an example of a chosen plaintext attacks.

- **Adaptively Chosen Plaintext-Ciphertext Attacks**: This attack scenario seems to be unrealistic but in terms of the security of the designed system it is still crucial. Here, the attacker is assumed to gather both the encryption and the decryption boxes and has full control over the plaintext and the ciphertexts. Boomerang attack is an example of adaptively chosen plaintext-ciphertext attacks.

- **Related-Key Attacks**: In the related-key attack scenario, it is again assumed that the only secret parameter is the key. However, this time there exist many keys and the attacker knows the relations between the keys (e.g. differences between the keys) but not the exact values of the keys. Combined attacks with related keys can be given as the example to the related-key attacks.

The most trivial method to break a cipher is the *exhaustive search* meaning to try all possible secret key combinations. However, by breaking a cipher, we mean to generate a method to distinguish the cipher from a random permutation or revealing some parts of the secret key faster than the exhaustive search. All attack methods introduced so far exemplify this simple trivial fact.

1.2 Hash Functions

Hash functions, more precisely cryptographic hash functions are one of the key primitives of cryptography which are being used in many areas such as unkeyed, symmetric or asymmetric cryptography. They are used widely in cryptographic applications such as digital signatures, information authentication, redundancy, protection of passwords, confirmation of knowledge/commitment, pseudo-random string generation and key derivation.

Cryptographic hash functions are used to map arbitrary length data to fix length *hash* (message digest, fingerprint) value. They are the digital fingerprints of the compressed messages and used to identify the original messages as in the real meaning of fingerprint. However, in cryptographic hash functions, it is highly crucial that given a hash value $h(x)$, it is hard to find the original message x that is compressed by the hash function h. Besides, effectiveness of the compression process is also very important for hash functions.

The most well known hash functions today assume the design principles of `MD4`[69] which is designed by Rivest. `MD5`[70], `SHA0`[62] and `SHA1`[61] are some of the examples of `MD4`-descendants where the last one is used as FIPS for secure hashing. Recently, many of these family have been broken by the extraordinary improvements in cryptanalysis of hash functions.

As the hash functions map arbitrary length data to a fix length hash value, it is trivial that there exist many collisions which is the basis of cryptanalysis of hash functions. Regarding this simple fact, some basic properties of a cryptographic hash function are listed below:

- **Preimage Resistant:** For a given value $h(x)$ and the compression function h, it should be 'hard' to compute x.
- **Second Preimage Resistant:** Given x and $h(x)$, it should be 'hard' to find x' such that $x \neq x'$ and $h(x') = h(x)$.
- **Collision Resistant :** For any x it should be 'hard' to find x' where $x \neq x'$ and $h(x) = h(x')$.

From the definitions given above collision resistance implies second preimage resistance.

In block cipher cryptanalysis, the theoretical bound for breaking a cipher is taken to be the complexity of the exhaustive search. However, in cryptanalysis of hash functions this is not the case because we are searching for colliding pairs instead of finding the secret key that matches plaintext-ciphertext pairs. The *birthday paradox* limit is taken as a basis for finding collisions for hash functions.

Birthday attack can be described as follows. Suppose that $h(x)$ is a random function where $h(x)$ is the set of all possible 2^n values. One expects a collision in about $2^{n/2}$ evaluations of h [26]. Therefore, if one is able to find a collision for a hash function (which is producing a hash value of 2^n bits) in less than $2^{n/2}$ evaluations of h , then h is assumed to be broken.

Let h be a hash function producing n bit fingerprint. Collision attacks to h can be divided in three major parts :

1. **Collision Attack :** In this scenario, the attacker tries to find at least one colliding pair in less than $2^{n/2}$ evaluations of h.
2. **Near-Collision Attack :** In near-collision attack, the attacker tries to find at least one pair whose hash values are same in many of the bits in less than $2^{n/2}$ evaluations of h. Near-collision is not a collision at all but it can lead to collisions for further message blocks.

3. **Pseudo-Collision Attack :** If $h(x)$ uses an Initial Value (IV) as a starting step, this IV is assumed to be fixed to some constants. In pseudo-collision attack, the attacker starts with free IVs instead of a fixed value and tries to find collision.

4. **Pseudo-Near-Collision Attack :** This is a free start near-collision attack. Namely, the attacker chooses IVs and tries to find near-collisions.

These are the basic attack scenarios for hash functions and that are used throughout the thesis. One of he common features of the latest attack scenarios to hash functions is to adopt differential cryptanalysis which will be introduced in the next chapter.

1.3 Our Contributions and The Structure of the Thesis

In this work, we consider mainly cryptanalysis of block ciphers and hash functions. Starting from the differential cryptanalysis, we extend the notion to the other block cipher attacks inspired from differential cryptanalysis, namely boomerang, amplified boomerang and rectangle attacks together with their related-key versions. Our aim is to adopt some block cipher attacks to hash functions to find collisions. In this respect, we take the well known block cipher based hash function `Tiger` as a starting point and we tried to mount these attacks on `Tiger`.

Our first contribution is to apply a related-key boomerang and rectangle attacks to the encryption mode of `Tiger`. We convert `Tiger` to a block cipher and found 17, 19 and 21-round distinguishers. These attacks are distinguishing attacks but can easily be converted to key recovery attacks. In addition, the exact values of the differences used in the attack of Kelsey and Lucks are found and the collision attack on `Tiger`-16 is slightly modified.

The structure of the thesis is as follows. In Chapter 2, we make a recapitulation of differential cryptanalysis and give some basic examples of differential cryptanalysis. In Chapter 3, the extensions of the differential cryptanalysis to boomerang, amplified boomerang and rectangle attacks together with their related-key combined attack versions are detailed. Moreover, the application of these attacks to the encryption modes of `MD4` and `MD5` is given. In Chapter 4, we give some of the previous collision attacks to `Tiger`. Firstly, we give the details of the collision attack of Kelsey and Lucks to reduced round `Tiger` with slight modifications and the pseudo-near-collision attack of Mendel and Rijmen to full `Tiger`. Our contributions are in Chapter 5. Namely, 17, 19 and 21-round distinguishers to the encryption mode of `Tiger` are presented. Finally, Chapter 6 concludes the thesis.

Chapter 2

Differential Cryptanalysis

This chapter is mainly dedicated to differential cryptanalysis which is known to be one of the generic, methodological and statistical block cipher attack. Its extensions to the boomerang attack, amplified boomerang attack, rectangle attack and their related-key combined attack versions will be mentioned in the following chapters.

Differential cryptanalysis which exploits the differential relations between the plaintext and ciphertext pairs is the first attack that breaks `DES` theoretically. Although it was claimed to be known, it was introduced in 1990 by Biham and Shamir in [13] by attacking reduced round `DES` and then improved to a full attack in [15]. Then, Knudsen extended the differential cryptanalysis to truncated and higher order differentials [47].

Many block ciphers known today are not vulnerable to differential cryptanalysis since a lot of work done on this subject. However, still it is the key primitive for block cipher designers to show that the designed cipher is secure against the differential cryptanalysis. Besides, the most powerful attacks to hash functions so as to find collisions known today are all differential attacks. Also, in recent years there are a lot of papers adapting differential cryptanalysis to the stream ciphers as well [86, 87, 59, 29]. Therefore, the resistance against differential cryptanalysis plays a crucial role in cryptography.

The overview of this chapter is as follows. The Section 2.1 introduces `DES` very briefly that is the first block cipher attacked by differential cryptanalysis. The pure differential cryptanalysis is discussed in Section 2.2. Sections 2.3 and 2.4 exemplify the differential cryptanalysis on `DES` and `FEAL`-8, respectively.

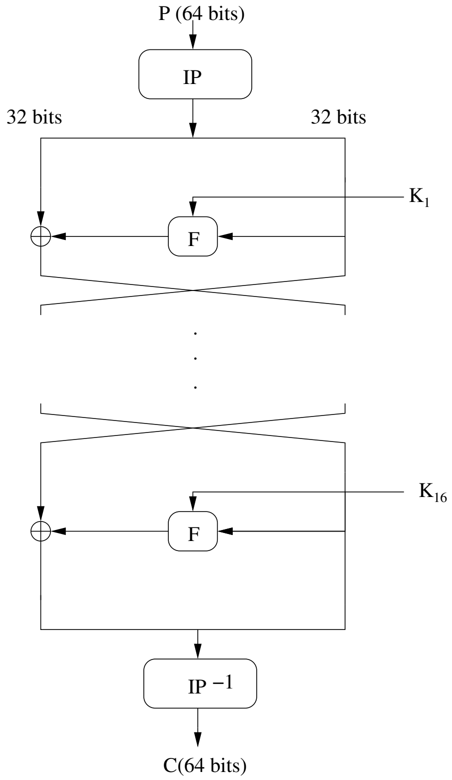


Figure 2.1: The General Overview of DES

2.1 DES

DES [63] is a block cipher which is selected as FIPS for the United States in 1976. It operates on 64 bits encrypting 64-bit plaintext to the 64-bit ciphertext using a relatively short 56-bit secret key. Nowadays, DES is believed to be insecure as there exist many attacks on DES. However, some new versions such as triple DES is still being used in many applications.

The history of DES commences at the early 1970s from a need for a government-wide standard for encrypting unclassified information. In 1976, the proposal from IBM found suitable for the DES and it has been used widely since then. Although it is claimed to be insecure against differential cryptanalysis in late 70s, until 1990 it is kept secret. In 1990, the first attack was applied on 15-round DES faster than exhaustive search by Biham and Shamir [13]. Two years later, this attack was extended to the full DES.

The general overview of DES is shown in Figure 2.1. It is a typical Feistel Cipher operating on 64 bits. The round function of DES takes 32-bit input and produces a

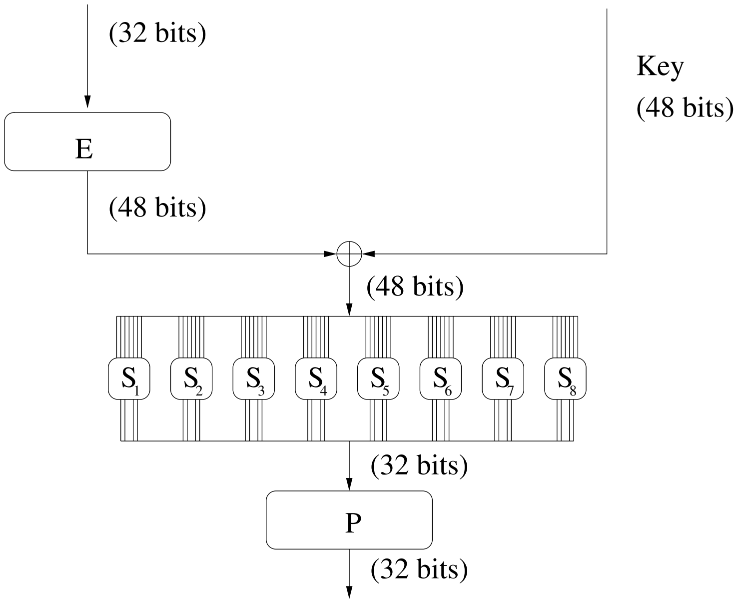


Figure 2.2: The Round Function of DES

32-bit output. 32-bit input is first expanded by E to 48-bit to be able to be XORed with 48-bit subkey. Then, 8 different 6×4 Substitution Boxes (S-Boxes) are used to provide the non-linearity and 32-bit output of the substitution layer. Finally, a permutation P is applied to the 32-bit data to produce 32-bit output of the round function. One round of DES is visualized in Figure 2.2. The exact tables for E, P and S-boxes can be found in the original proposal [63]. For the time being, we exclude the key scheduling algorithm as it is not directly related to the pure differential cryptanalysis.

2.2 The Overview of the Attack

Differential cryptanalysis is a chosen plaintext attack and proposed for block ciphers at first. However, the straightforward generalizations can be easily be made to stream ciphers and hash functions as well. Given two plaintexts P_1 and P_2 with a predetermined difference, the attacker tries to exploit some expected differences between their ciphertexts C_1 and C_2 with a high probability. These type of attacks are called as *distinguishing attacks* and used to distinguish the cipher from a random permutation. For the key recovery attacks, on the other hand, the attacker tries to exploit the differences between the outputs of the rounds ($1, 2$ rounds before the ciphertext depending on the cipher) before the ciphertexts.

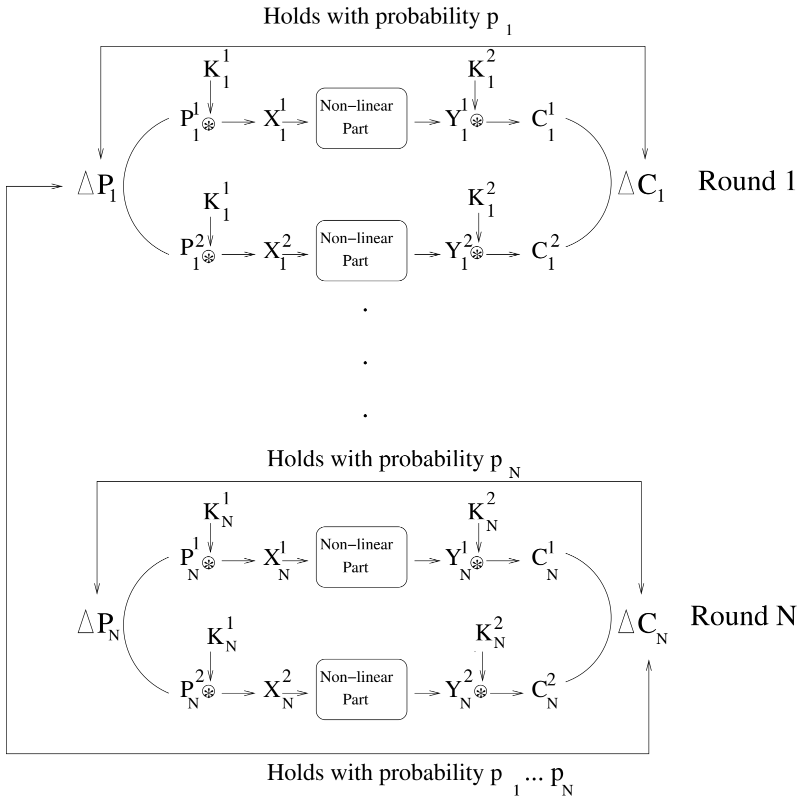


Figure 2.3: Overview of the Differential Cryptanalysis

The most attractive part of the differential cryptanalysis is the fact that it makes an extensive use of ignoring the key effect which is the only secret parameter of the cipher. That is, the differences between the plaintexts are arranged according to the operations used in the round functions to cancel the key part as just one key is used for both of the plaintexts. Let $\circledast$ be the operation to combine the data P with the key K (that is, $P \circledast K$). Then the difference between the two data are chosen as:

$$\Delta(P_1, P_2) \quad = \quad P_1 \circledast P_2^{-1}.$$

So, the difference after key addition is

$$\Delta(P_1 \circledast K, P_2 \circledast K) \quad = \quad P_1 \circledast K \circledast K^{-1} \circledast P_2^{-1} = \Delta(P_1, P_2)$$

Therefore, the key effect through the differences can be discarded by this simple trick. In general, XOR or modular addition are used as $\circledast$ and the inverse of a data can be found easily. For an R round random cipher operating on n bits, the probability of expecting the corresponding ciphertext difference is 2^{-n}. However,

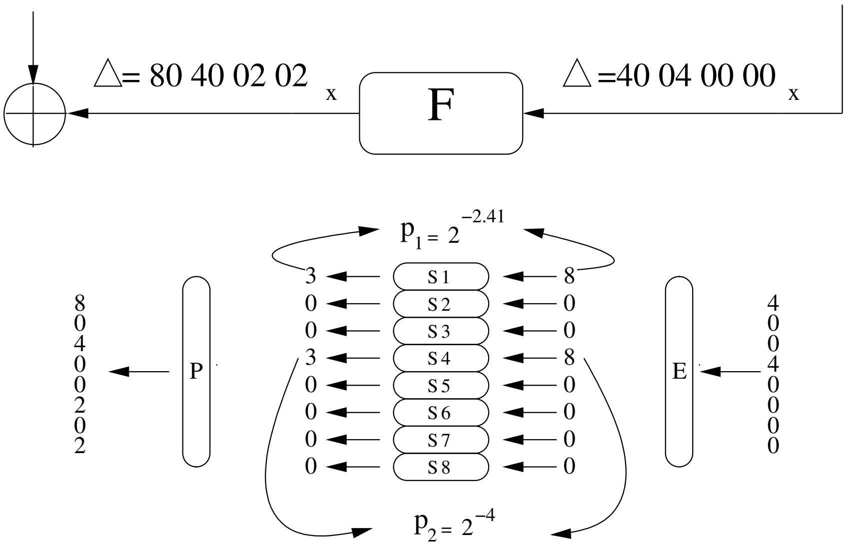


Figure 2.4: A One Round Differential for `DES`

differential cryptanalysis works under the fact that given a plaintext difference, the corresponding ciphertext difference occurs with a probability much higher than 2^{-n}.

In each round, the corresponding input difference and the expected output difference, together with the probability p of this expectation, are gathered and the so called *the differential* $(\Delta P, \Delta C, p)$ is constructed. The non-linear parts of the round functions, such as S-boxes, affect the probability of the differential at each round. Assuming the independence, the total probability is calculated as the multiplication of the differential probabilities of each round. The model is visualized in Figure 2.3.

In the round function components, the difference propagates linearly through the linear operations as permutations or expansions. However, the nonlinear components such as S-boxes do not let the difference propagate linearly. *The Difference Distribution Table* (DDT) or *The XOR Table* of an S-box is defined to extract the probabilities of getting some specific output difference given the input difference. The i^{th} row and j^{th} column of a DDT correspond to the input difference and the output difference respectively. The intersection shows the number of occurrences of the corresponding differences.

Figure 2.4 shows a one round differential characteristic for `DES` [28]. 40 04 00 00_x input difference enters to the round function F and 30 03 00 00_x output difference is expected with probability $2^{-4.1}.2^{-4} = 2^{-8.1}$. The input difference propagates linearly through the expansion E and the permutation P. However, the first and the fourth S-boxes are active and both seek for the same input and the output differences; namely

8_x and 3_x respectively.

For the key recovery attack, the differential characteristic is set up to a number of rounds before the last round. That is, depending on the cipher (Feistel or SP Network), in order to recover the key, the differential characteristic must be constructed up to a reference round ($1, 2$ rounds before the last round). Afterwards, by guessing the subkeys of the remaining rounds and decrypting the ciphertexts up to that reference round, the attacker is able to analyze the differential characteristic to hold with some prescribed probability.

The guessed subkey bits are called *target partial subkeys* and fully determined by the differential characteristic and the differences in the reference step. Let us assume for a moment that the differential characteristic starts from the first round (some tricks can be used as *structures* to discard first round as in the differential attack to full round `DES` [15]). The Figure 2.5 shows the differential key recovery attack in detail where the first two are the classical Feistel Networks and the third one is a basic Substitution Permutation Network. The last round subkey K_R is guessed for all three of the ciphers and the corresponding ciphertext pairs are decrypted until the reference step shown with a dashed horizontal line. Another dashed line with arrows shows the known part of the ciphertext pairs through the decryption process which enable the attacker to get the differences whereever necessary.

The number of needed pairs generally depends on the characteristic probability, the number of subkey bits guessed and on the level of identification of the right pairs [14]. A *right pair* is defined to be the pair of plaintexts together with their ciphertexts that obey to the prescribed differential path. A *noise*, on the other hand, is the plaintext-ciphertext pairs that obey the differences in their plaintexts and ciphertexts but not obey the prescribed characteristic. It is trivial that for a characteristic with probability p, one needs at least $1/p$ pairs to distinguish the cipher from a random permutation. In fact, the number of pairs is taken to be c/p, where c is a constant dependent to the so called S/N(Signal to Noise) ratio which is known to be the ratio of the probability of the right key being suggested by a right pair to the probability of a random key being suggested by a random pair with the given initial difference:

$$S/N = \frac{2^k . p}{\alpha . \beta}$$

Here, k is the number of active bits, p is the characteristic's probability, α is the

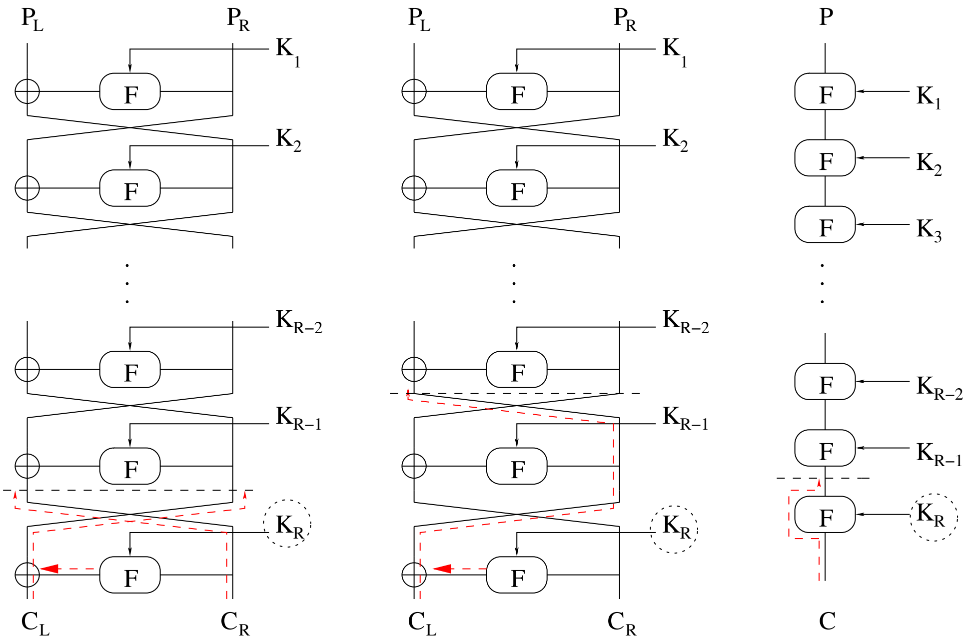


Figure 2.5: The Differential Key Recovery Attack for Feistel Networks and SPNs

number of keys suggested by each pair of plaintexts and β is the fraction of the counted pairs among all pairs [14](For details refer to [13, 15, 14]). When S/N is sufficiently larger than 1, only a few pairs are needed for the attack. If S/N ratio is closer to 1, then more data is needed to distinguisher the cipher from a random permutation (If S/N ratio is less than one the attack can not be applied).

The following subsections contain two examples of the differential attacks: Differential Cryptanalysis of `DES` and `FEAL`-8. The former is the first example in the literature and the latter is different in the sense that it uses an efficient algorithm to calcute the differential probabilities of modular addition over XOR and does not make use of the tricky points used for `DES`. Differential cryptanalysis of `FEAL`-8 is an elementary application that is detailed more by constructing the differential characteristics step by step.

2.3 Differential Cryptanalysis of `DES`

The application of the differential cryptanalysis is not a straightforward manner since it is difficult to construct the differential characteristic. There are some tricky points to extend the one-round characteristics to the whole cipher, one of which is the use of *the iterative characteristics* that enable to iterate the characteristics as many times

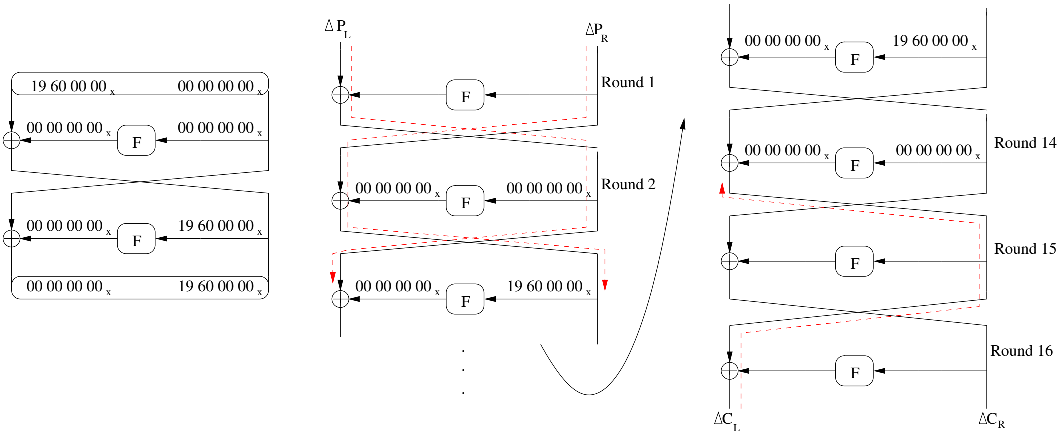


Figure 2.6: The Differential Characteristics Used for `DES`

as needed. Moreover, given a non-zero input difference, searching for a zero difference in the output difference with high probability is another tricky point to construct the differential characteristic. Both are used to attack `DES` [13, 15].

In the attack, a one-round differential with zero output difference of probability 1/234 is combined with the trivial characteristic making a 2-round iterative differential characteristic. It is iterated 6.5 times to construct the 13-round characteristic with probability $(1/234)^6 = 2^{-47.2}$. This 13-round differential characteristic is used between the rounds 2 and 14. Finally, by going backwards up to 14^{th} round as described earlier, this characteristic can be used to attack `DES`. However, the first round needs to be modified for the characteristic. These characteristics are detailed in Figure 2.6.

In order to extend the attack to 16 rounds Biham and Shamir introduced the so called *structures* in [15]. Actually, the structures can be understood as the carefully chosen plaintexts. The aim here is to use structures such that without loss of probability the required plaintext pairs with the prescribed difference are generated before the second round. Let P be an arbitrary plaintext and $c_1, ..., c_{4096}$ be the all possible output differences of the S-boxes S_1, S_2 and S_3 leading to zero differences in remaining 20 bits. The structure of 2^{13} plaintexts are defined to be:

$$P_i = P \oplus (c_i, 0) \quad , \quad P_i^* = P_i \oplus (0, 19600000_x) \text{ , for } 1 \leq i \leq 2^{12}$$
$$C_i = \text{DES}(P_i) \quad , \quad C_i^* = \text{DES}(P_i^*)$$

Here one can generate 2^{24} plaintext pairs (P_i, P_j^*) with the XOR difference $\Delta =$

$(c_k, 19600000_x)$ $(1 \le i, j, k \le 2^{12})$ as:

$$\begin{array}{cccc} (P_1, P_1^*) \ , \ (P_2, P_1^*) \ , & \dots & , (P_{2^{12}}, P_1^*) \\ (P_1, P_2^*) \ , \ (P_2, P_2^*) \ , & \dots & , (P_{2^{12}}, P_2^*) \\ & \vdots & \\ (P_1, P_{2^{12}}^*) \ , \ (P_2, P_{2^{12}}^*) \ , & \dots & , (P_{2^{12}}, P_{2^{12}}^*) \end{array}$$

Obviously in 2^{24} pairs above each $\Delta(P_1, P_j^*), \Delta(P_2, P_j^*), \dots, \Delta(P_{2^{12}}, P_j^*)$ is one of the constants $(c_k, 19600000_x)$. Thus, each c_k appeared 2^{12} times in above pairs. As a result, one can cancel the differences in the output of the round function of the first round exactly for 2^{12} plaintext pairs. Therefore, this leads to offset the first round by preparing necessary plaintexts for the differential characteristic starting from the second round. Now, for each structure the probability of holding the differential characteristic is $2^{12}.2^{-47.12} = 2^{-35.12}$.

Differential cryptanalysis of DES is an important example of the use of iterative characteristics and the structures. Differential cryptanalysis of FEAL, on the other hand, can be given as an important example of the use of high probability characteristics and extending this characteristics to many rounds.

2.4 Differential Cryptanalysis of FEAL-8

2.4.1 FEAL-8

FEAL-8 [58] is a DES-like block cipher and was designed to be as secure as DES but faster than DES in many platforms. It is an 8 round cipher with 128-bit key length and 64-bit block length. It can in fact be broken by many type of attacks because its simplicity in the round function. The general overview of FEAL-8 is given in Figure 2.7. Let us ignore the input and output whitening operations for a moment as it can be discarded easily (for further details refer to [14]).

FEAL-8 has a very simple round function which is depicted in Figure 2.7. From the differential cryptanalysis point of view, one really needs to be careful about the simplicity of S-Boxes. The non-linearity of S-Boxes used in FEAL-8 heavily depends on the modular addition *mod* 256. The reason why the designers of FEAL-8 have chosen those S-Boxes is the speed of the addition operation. Actually, it is difficult to analyze and read the 2^{24} entries in the XOR table of FEAL-8. Therefore, there is a

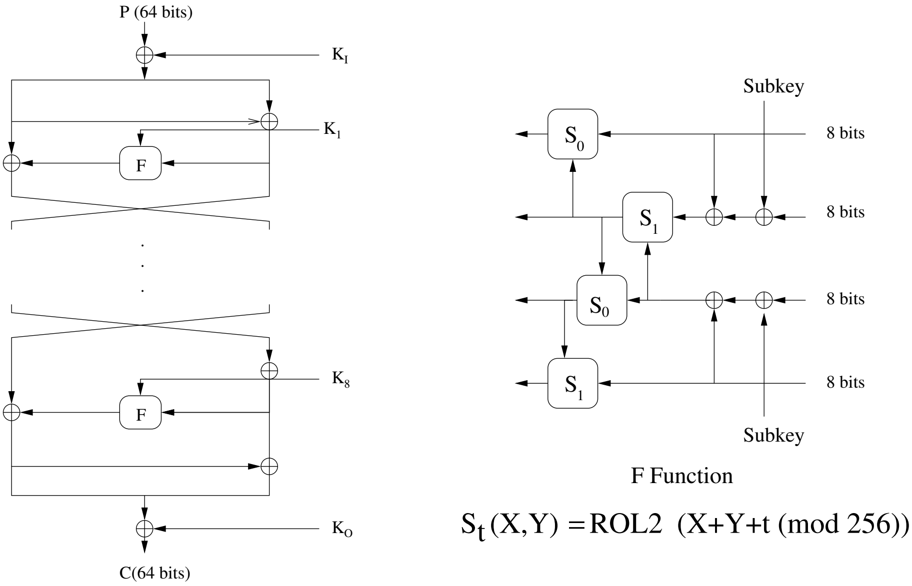


Figure 2.7: `Overview FEAL-8`

need to find the effect of modular addition on XOR operation instead of investigating the XOR Tables. In [51] Lipmaa and Moriai makes it very easy to construct such probabilities by introducing efficient algorithms for differential probability of addition mod 2^n that will be summarized in the following subsection.

2.4.2 Efficient Algorithm for Computing Differential Properties of Addition

Originally, differential cryptanalysis was considered with respect to XOR and then generalized to an arbitrary group operation. Until now it has seemed that the problem of evaluating the differential properties of addition with respect to XOR is hard and it is used in many ciphers to obtain non-linearity very efficiently. The differential probability of addition with respect to XOR can be defined as:

$$DP^{+}(\alpha, \beta \to \gamma) := P_{x,y}[(x + y) \oplus ((x \oplus \alpha) + (y \oplus \beta)) = \gamma]$$

The fastest known algorithms for computing the differential probability of addition DP^{+} is exponential in n [51]. However, in [51] Lipmaa and Moriai present a log-time algorithm for DP^{+}. Let us introduce some brief notation together with some auxiliary conventions.

Let $\Sigma^n = \{0, 1\}$ be the binary alphabet. For any n-bit string $x \in \Sigma^n$, x_i be the i^{th}

coordinate of x (i.e., $x = \sum_{i=0}^{n-1} x_i 2^i$). We always assume that $x_i = 0$ if $i \notin [0, n-1]$. Let $\oplus, \vee, \wedge$ and $\neg$ denote n-bit bitwise XOR, OR, AND and negation, respectively. Let $x >> i$ (resp. $x << i$) denote the right (resp. the left) shift by i positions. Addition is always performed modulo 2^n, if not stated otherwise.

For any x,y and z we define

$$\begin{aligned} eq(x,y,z): &= (\neq x \oplus y) \wedge (\neq x \oplus z) \\ \text{that is, } eq(x,y,z)_i &= 1 \Leftrightarrow x_i = y_i = z_i \text{ and} \\ xor(x,y,z): &= x \oplus y \oplus z \end{aligned}$$

For any n, let $mask(n) := 2^n - 1$. We say that differential $\delta = (\alpha, \beta \rightarrow \gamma)$ is good if

$$eq(\alpha << 1, \beta << 1, \gamma << 1) \wedge (xor(\alpha, \beta, \gamma) \oplus (\alpha << 1)) = 0$$

For the detailed description of the algorithm refer to [51]). The following algorithm gives the desired value for the differential property of the addition [51].

Algorithm 2.4.1: A LOG-TIME ALGORITHM FOR $DP^{+}((\alpha, \beta \rightarrow \gamma))$

---

if $eq(\alpha << 1, \beta << 1, \gamma << 1) \wedge (xor(\alpha, \beta, \gamma) \oplus (\beta << 1)) \neq 0$
return (0);
return $(2)^{-w_h(\neg eq(\alpha,\beta,\gamma) \wedge mask(n-1))}$

It is obvious that this algorithm is very simple. Special functions used in the algorithm can be investigated by the original paper. Now, let us construct the round characteristics. For the remaining part of this section the notation $(\Delta x, \Delta y) = \Delta z$ is used to denote the input-output XOR differences.

2.4.3 Constructing Differential Characteristics

Using the above algorithm or by just observing, it is easy to see that there is no difference if we change the places of Δx and Δy. Also, it is obvious that the below differences are satisfied with trivial probability because of the simple operation used

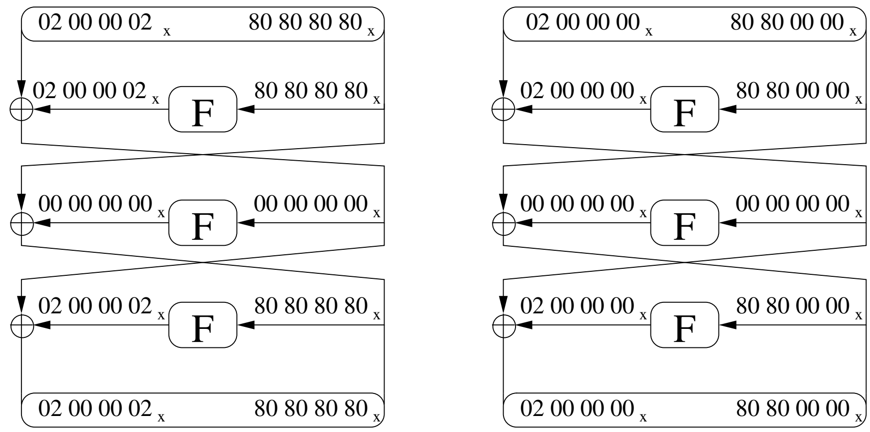


Figure 2.8: 3-Round Characteristics of FEAL-8

in the S-boxes of FEAL-8 (The differences are written in hexadecimal).

$$(\Delta x = 00, \Delta y = 00) = \Delta z = 00$$
$$(\Delta x = 80, \Delta y = 80) = \Delta z = 00$$
$$(\Delta x = 80, \Delta y = 00) = \Delta z = 02$$

In fact, it is good to see that 3 such one-round characteristics are found because many round characteristics with $p = 1$ can be constructed by using these one-round characteristics. During the attack, we make an extensive use of this fact. In order to construct a powerful characteristic, the above characteristics should be used as much as possible. Transition from a 1-round characteristic to 3-round characteristics is straightforward and shown in Figure 2.8.

The aim here is to extend these characteristics as much as possible. Therefore, by using 3-round characteristics, 5-round characteristics can be constructed similarly. However, this time a 1-round characteristic is needed for the second and the fourth round. The strategy that we follow is to take the obvious characteristic in the middle, namely in the third round and to arrange the first and the last round according to the middle. Let us investigate the 5-round characteristic given in Figure 2.9. As seen, the rounds $1, 3$ and 5 are satisfied with the maximum probabilities. If the most effective differences are chosen for the rounds 2 and 4, a 5-round characteristic will be obtained. As seen from the figure, the new aim now is to find the input difference which corresponds to the output difference (80 80 00 00) with high probability. In

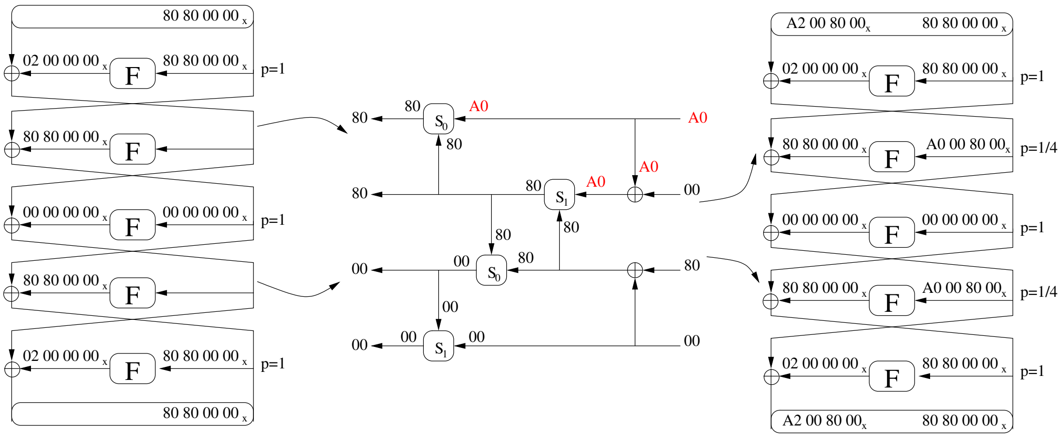


Figure 2.9: 5-Round Characteristic of `FEAL-8`

order to find such a characteristic, the round function F should be investigated carefully to find the most efficient value.

As seen from Figure 2.9, the highest probability and the input difference Δx which corresponds to $(\Delta x, \Delta y = 80) = \Delta z = 80$ should be found. With the help of the algorithm given above, by trying 256 possible combinations, $x = A0$ input difference is found to give the highest probability with $p = 1/2$. It is really an efficient probability and when applied to 2 S-Boxes in the F-function a new one round characteristic can be found working with probability 2^{-2}.

The last step to finalize the attack is to extend the 5-round characteristic to 6-round characteristic. This time the process followed in the construction of the 5-round characteristic will be repeated for the 6-round characteristic for the round function. The characteristic shown in Figure 2.10 works with probability $p = 1/16$ and sufficient for the time being. Moreover, it is the simplest way to extend the 5-round characteristic. What needs to be done is to find the most efficient output difference which corresponds to the input difference ($A2$ 00 80 00). Therefore, F-function should be investigated again.

The Figure 2.10 summarizes the 6-round differential characteristic search. During the process, it is made an extensive use of the algorithm given above. The 6-round characteristic works with probability 1/256 and it is a really good characteristic in order to break the cipher because a few plaintext-ciphertext pairs are needed.

We simulated with characteristic probability $p = 1/128$, excluding one active S-Box in the last round. 2000 chosen plaintexts are collected and the attack is mounted. The last round subkey can be found in a few seconds uniquely. We expected to have

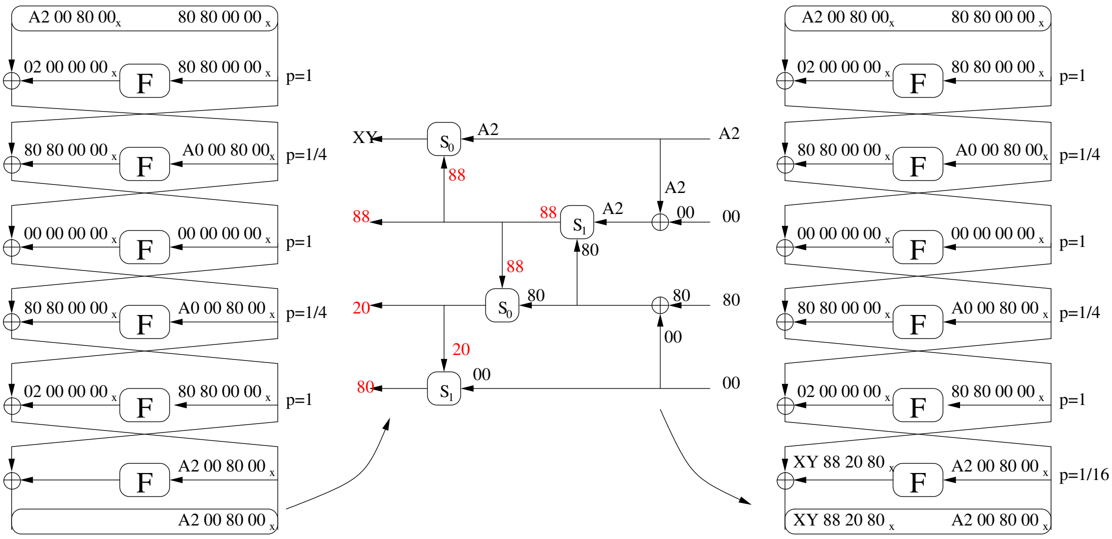


Figure 2.10: 6-Round Characteristic of `FEAL-8`

$15-16$ pairs to satisfy the above characteristic for the right pair and we found that at least 13 pairs satisfying the characteristic.

2.5 Resistance Against Differential Cryptanalysis

As its name suggests this section is devoted to discuss the security of a cipher against the differential cryptanalysis. A cipher's security against differential cryptanalysis is very important in that it is assumed to be the first and vital step for a designer to prove the security of the designed cipher. Let us assume that the confusion is gathered through S-boxes for simplicity (for the other type of confusion mechanisms as in `IDEA`, the attacker is assumed to calculate the differential probabilities up to some reference point, e.g. the Hamming Weight of the input differences are upper bounded by some constants).

The security of the cipher against differential attacks is not generally shown by finding some round characteristics. Instead, some tight complexity bounds are given by considering the number of active S-boxes. An *inactive S-box* is defined to take a zero input difference. Once an S-box is active, it is assumed that the differential holds with DP_{max}. We assumed that the confusion layer is gathered through S-boxes because we are able to construct the XOR Tables and the maximum probability $\mathrm{DP}_{\mathrm{max}}$ in XOR table.

Since we are searching for the best differential, if we have at least n active S-boxes, then one can conclude that the probability of any differential characteristic can be at most $\mathrm{DP}^n_{\mathrm{max}}$. Therefore, one can easily prove that if the designed cipher fulfills this complexity bound, then it is resistant against differential cryptanalysis. Namely, if DP^{-n}_{max} exceeds the exhaustive search bound, then one can conclude that the designed cipher is secure against differential cryptanalysis. This is given in terms of the number of rounds attacked.

This type of analysis is not practical because the true complexity of mounting a practical differential attack is often much higher than the bound indicates. Nonetheless, when $\mathrm{DP}^n_{\mathrm{max}}$ is sufficiently small, the results can provide powerful evidence of the ciphers' security against differential cryptanalysis.

In the folowing subsection, a recently designed cipher `CLEFIA`'s [78] security against differential cryptanalysis will be discussed.

2.5.1 An Example: `CLEFIA`

`CLEFIA` [78] is a new block cipher introduced in FSE '07 by Shirai *et al.* It is a relatively fast and modern cipher and the key sizes can be chose as 128-bit, 192-bit or 256-bit. It is a generalized unbalanced Feistel Scheme with four branches and the round number depends on the key length (18, 20, 22 rounds for 128, 192 and 256 bit keys respectively). The general overview of `CLEFIA` is given in Figure 2.11.

`CLEFIA` uses two types of round functions F_0 and F_1 as depicted in Figure 2.12. In the round function, firstly 32-bit input is XORed with the subkey and enters to the substitution level. In `CLEFIA` two 8×8 S-boxes S_0 and S_1 are used and they are applied alternatively in F_0 and F_1 but in different order. Finally, again two different Maximum Distance Separable (MDS) codes are used as the diffusion layer (Again we omit the key scheduling algorithm). The detailed description of `CLEFIA` is given in [78].

`CLEFIA` is shown to be secure for all known cryptanalytic attacks in the proposal. Resistance of `CLEFIA` against differential cryptanalysis is proved by the analysis given in previous subsection. The maximum probability in DDT of both S-boxes are gathered and the number of minimum active S-boxes is calculated for `CLEFIA`. The Table 2.1 shows the number of rounds and the minimum number of active S-boxes of `CLEFIA` under 128, 192 and 256-bit keys where the number of rounds are 18,22 and 26, respectively and $\mathrm{DP}_{\mathrm{max}} = 2^{-4,67}$. For 128-bit key version, since $(4.67) \times 28 = 130.76 > 128$, it can be proved theoretically that reduced 12-round `CLEFIA` is secure against differ-

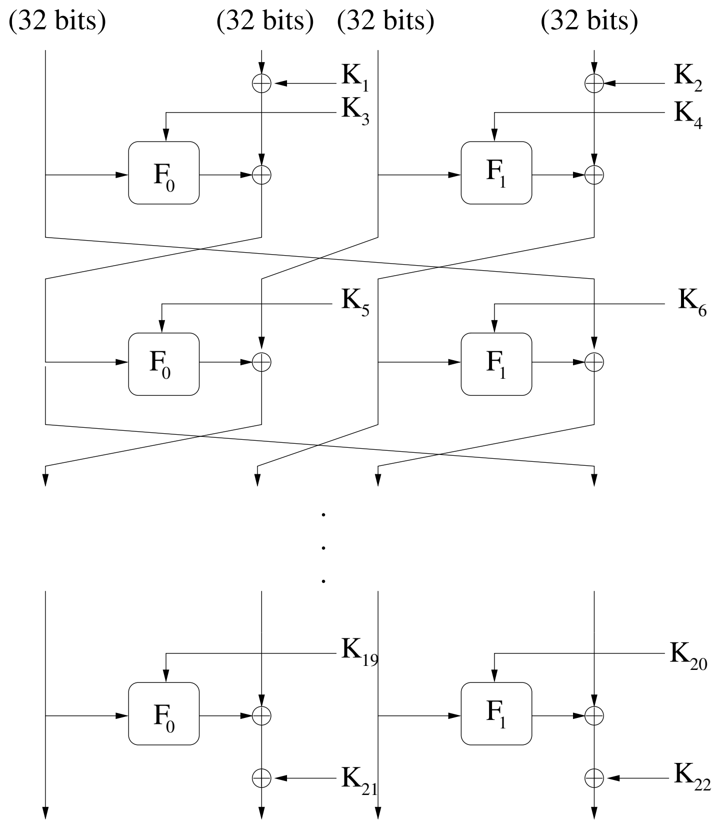


Figure 2.11: CLEFIA Block Cipher (128-bit key version)

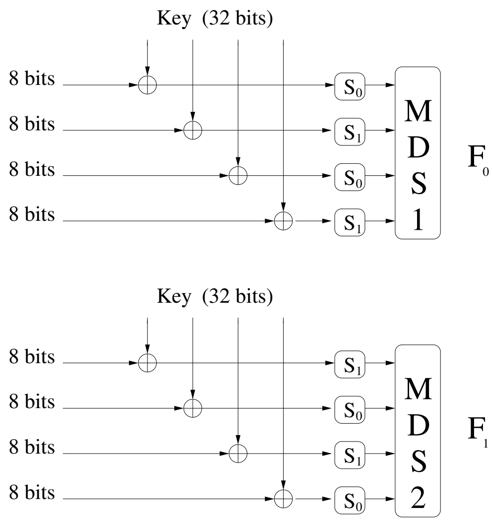


Figure 2.12: CLEFIA Round Function

Round	Number of Active S-boxes	Round	Number of Active S-boxes
1	0	14	34
2	1	15	36
3	2	16	38
4	6	17	40
5	8	*18	44
6	12	19	46
7	14	20	50
8	18	21	52
9	20	*22	55
10	22	23	56
11	24	24	59
12	28	25	62
13	30	* 26	65

Table 2.1: Number of Active S-boxes of `CLEFIA`

ential cryptanalysis. Similar calculations can be done for other key lengths.

Chapter 3

Extensions of the Differential Cryptanalysis

This chapter is devoted to the extensions of the differential cryptanalysis to the other block cipher attacks inspired from differential cryptanalysis. Throughout this chapter, the boomerang, amplified boomerang and rectangle attacks together with their related key combined attack versions are mentioned.

While early 1990s witnessed the pure differential cryptanalysis and its slight improvements, it was shown in late 1990s and early 2000s that differential cryptanalysis can further be improved to a new type of attacks; namely the combined attacks which were first exemplified by the differential-linear attacks in [50]. The boomerang attack was introduced in [80] as the extension of the differential-linear attack to differential-differential attack that effectively makes use of the two short differential characteristics instead of one long one. Afterwards, the boomerang attack was improved to amplified boomerang attack and the rectangle attack in [37] and [8], respectively.

Nowadays, on the other hand, the boomerang attacks and the rectangle attack have been applied together with the related-key model [5]. That is, the new type of combined attacks called related-key boomerang and rectangle attacks [9, 11, 12, 44] are two of the most effective block cipher attacks. The best attacks applied to `AES` [43, 12] known today are of these type.

This chapter is structured as follows. Section 3.1 introduces the boomerang and the related-key boomerang attack. Then, the boomerang attack is extended to amplified boomerang and rectangle attacks together with their related-key combined attack versions in Section 3.2 and 3.3. In Section 3.4, the application of these attacks to the encryption modes of `MD4` and `MD5` is presented.

3.1 The Boomerang and The Related-Key Boomerang Attack

The Boomerang Attack [80] may be seen as the refinement or the effective use of the pure differential cryptanalysis. After the application of differential-linear cryptanalysis [50], the boomerang attack can also be called differential-differential cryptanalysis. In the boomerang process, instead of using one long-ineffective (low probability) differential, the attacker uses two short-high probability differentials to increase the number of rounds attacked and the probability of the differential. The disadvantage of the boomerang attack is its adaptively chosen plaintext-ciphertext nature. Namely, besides the encryption box of the attacked cipher, it is assumed to have the decryption box.

For the sake of simplicity, we will use the same notation as in [12]. The Boomerang Distinguisher treats the attacked cipher E as a cascade of two sub-ciphers E_0 and E_1 (E_i^K stands for encryption with key K), i.e. $E = E_1 \ o \ E_0$. As mentioned above, two short-high probability differentials are used, one for E_0 and one for E_1, in order to increase the probability of the distinguisher. Let $\alpha \rightarrow \beta$ with probability p be the first differential used for E_0 and $\gamma \rightarrow \delta$ with probability q be the second differential used for E_1. Notice that, once the differential is chosen in one direction, the same differential holds for the opposite direction. Namely, the differentials $\beta \rightarrow \alpha$ for E_0^{-1} and $\delta \rightarrow \gamma$ for E_1^{-1} hold with probabilities p and q respectively. The key step in the boomerang distinguisher is to combine these two differentials. The boomerang distinguisher works as follows:

- Take a randomly chosen plaintext P_1 and form $P_2 = P_1 \oplus \alpha$.
- Obtain the corresponding ciphertexts $C_1 = E(P_1)$ and $C_2 = E(P_2)$ through E.
- Form the second ciphertext pair by $C_3 = C_1 \oplus \delta$ and $C_4 = C_2 \oplus \delta$.
- Obtain the corresponding plaintexts $P_3 = E^{-1}(C_3)$ and $P_4 = E^{-1}(C_4)$ through E^{-1}.
- Check $P_3 \oplus P_4 = \alpha$.

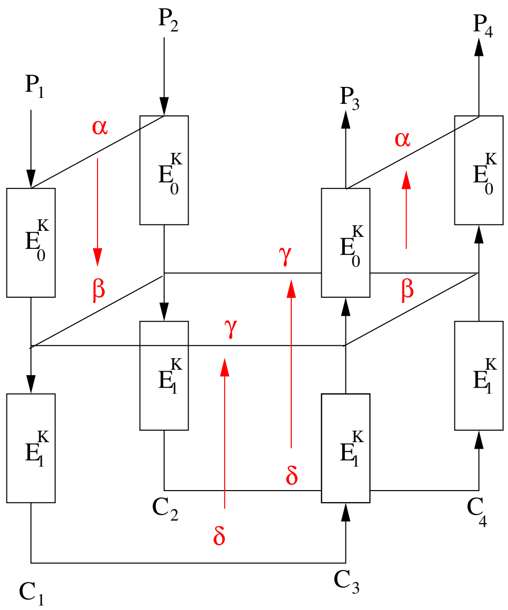


Figure 3.1: The Boomerang Distinguisher

After the first step of the above algorithm, the probabilistic arguments take place. While obtaining C_1 and C_2, we expect that the differential $\alpha \rightarrow \beta$ holds with probability p for E_0 once. There are no arguments about E_1 yet. Then, after the third step, through the decryption process we expect the differential $\delta \rightarrow \gamma$ holds with probability q for E_1^{-1} twice as we go backwards twice, once for each of the pairs (C_1, C_3) and (C_2, C_4). The crucial step of the boomerang distinguisher comes to the picture here when we are going backwards. Once we get $E_1^{-1}(C_3) \oplus E_1^{-1}(C_4) = \beta$, we are almost done as we know $E_0^{-1}(E_1^{-1}(C_3)) \oplus E_0^{-1}(E_1^{-1}(C_4)) = P_3 \oplus P_4 = \alpha$ holds with probability p. Now, it is time to explain how this is obtained.

$$
\begin{aligned}
E_1^{-1}(C_3) &\oplus E_1^{-1}(C_4) = \\
E_1^{-1}(C_3) &\oplus E_1^{-1}(C_4) \oplus E_1^{-1}(C_1) \oplus E_1^{-1}(C_1) \oplus E_1^{-1}(C_2) \oplus E_1^{-1}(C_2) = \\
E_1^{-1}(C_1) &\oplus E_1^{-1}(C_3) \oplus E_1^{-1}(C_2) \oplus E_1^{-1}(C_4) \oplus E_1^{-1}(C_1) \oplus E_1^{-1}(C_2) = \\
\gamma &\oplus \gamma \oplus E_1^{-1}(C_1) \oplus E_1^{-1}(C_2) = E_0(P_1) \oplus E_0(P_2) = \beta
\end{aligned}
$$

Therefore, the boomerang distinguisher works with probability p^2q^2. On the other hand, for a random permutation, the last step of the above argument holds with probability 2^{-n} where n is the number of the bits of each plaintext P. Thus, $pq > 2^{-n/2}$ must hold for the boomerang distinguisher. The boomerang distinguisher is visualized in Figure 3.1.

The related-key attack was first introduced by just considering rotational related-keys of `LOKI` [4]. Then, it was extended to differential-related-keys to attack many ciphers [39]. In differential related-keys, two differentials are combined; once for the standard encryption algorithm and once for the key scheduling algorithm. Namely, the standard differential model tries to increase $P(E_K(x) \oplus E_K(x \oplus \Delta x) = \Delta y)$. The related-key model, on the other hand, tries to increase $P(E_K(x) \oplus E_{K \oplus \Delta K}(x \oplus \Delta x) = \Delta y)$. The related-key boomerang attack, on the other hand, is one of the effective combined attacks on block ciphers that can be applied to many known block ciphers. For the related-key model, attacker assumes to know the relation (difference) between the keys, but not the exact values of keys.

The adaptation of related-key model to the boomerang attack is straightforward. The usual related-key model is applied to the subciphers E_0 and E_1 separately and the normal procedure is applied for the boomerang distinguisher. However, some additional properties are adapted for the related-key boomerang distinguisher. Instead of one pair of related-keys, 4 (or more) [43, 28, 40] related keys can be used and the most effective one is selected for the attack according to the structure of the cipher. We are going to give details about the related-key boomerang distinguisher based on 4 related-keys (as it is used in following chapters) as follows which is also shown in Figure 3.2:

- Take a randomly chosen plaintext P_1 and form $P_2 = P_1 \oplus \alpha$.
- Obtain the corresponding ciphertexts $C_1 = E_{K_1}(P_1)$ and $C_2 = E_{K_2}(P_2)$ through E, where $K_2 = K_1 \oplus \Delta K_{12}$.
- Form the second ciphertext pair by $C_3 = C_1 \oplus \delta$ and $C_4 = C_2 \oplus \delta$.
- Obtain the corresponding plaintexts $P_3 = E^{-1}_{K_3}(C_3)$ and $P_4 = E^{-1}_{K_4}(C_4)$ through E^{-1}, where $K_3 = K_1 \oplus \Delta K_{13}$, $K_4 = K_3 \oplus \Delta K_{12}$.
- Check $P_3 \oplus P_4 = \alpha$

The probabilistic arguments are same as in the boomerang distinguisher but they are converted to the related-key model for the related-key boomerang distinguisher. However, the non-linearity of the key scheduling algorithm is very important. It is not guaranteed in the boomerang distinguisher that the difference of the keys hold with high probability. The attacker should consider the probabilities carefully.

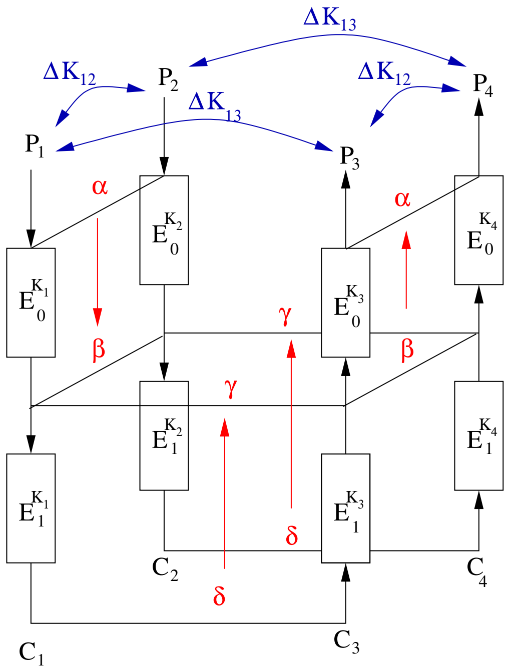


Figure 3.2: Related-Key Boomerang Distinguisher Based on Four Related Keys

Nonetheless, we are going to investigate the trivial differences caused by the key scheduling algorithm of `Tiger`.

3.1.1 Boomerang Attack on FEAL-8

In previous chapter, two 3-round trivial differentials were given for FEAL-8. Here, we show how to extend these characteristics immediately to 6-round by using the trivial 6-round boomerang distinguisher and improve the probability of the distinguisher immediately. Recall that the extension of the 3-round characteristics to 6-round characteristic for the standard differential cryptanalysis of FEAL-8 led to a loss of probability from one to 2^{-7}. Here, we show the benefits of using two short high probability differentials instead of long ineffective one. Figure 3.3 shows two trivial 3-round characteristics for FEAL-8.

It does not matter which of the characteristic matches with E_0 or E_1. Without loss of generality, let us assume for E_0 that the characteristic

$$\alpha = (02000002_x, 80808080_x) \longrightarrow \beta = (02000002_x, 80808080_x)$$

holds with probability one. Recall that the characteristic $\beta \longrightarrow \alpha$ also holds for E_0^{-1}

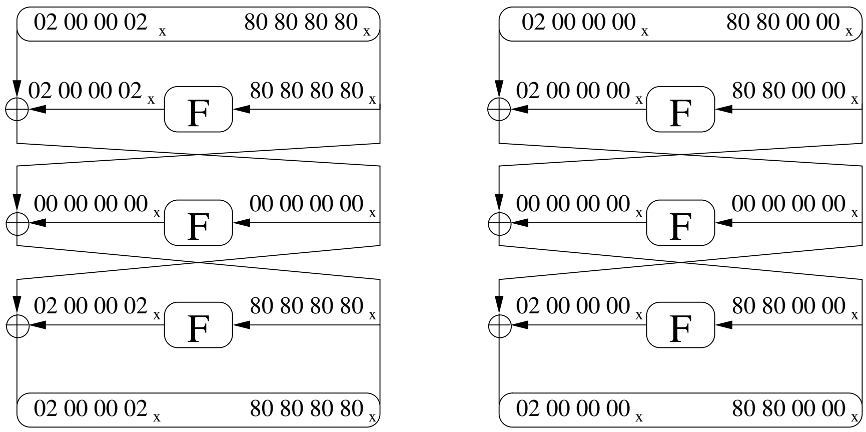


Figure 3.3: 3-Round Characteristics of `FEAL-8`

with the same probability. For the second subcipher E_1, the characteristic

$$\gamma = (02000000_x, 80800000_x) \longrightarrow \delta = (02000000_x, 80800000_x)$$

also holds with probability one.

The 6-round boomerang distinguisher works as follows for `FEAL`-8.

- Take a randomly chosen plaintext P_1 and form $P_2 = P_1 \oplus (02000002_x, 80808080_x)$.
- Obtain the corresponding ciphertexts $C_1 = E(P_1)$ and $C_2 = E(P_2)$ through E. At the end of this step we expect that the characteristic $\alpha \longrightarrow \beta$ holds for E_0 with probability one.
- Form the second ciphertext pair by $C_3 = C_1 \oplus \gamma = (02000000_x, 80800000_x)$ and $C_4 = C_2 \oplus \gamma = (02000000_x, 80800000_x)$.
- Obtain the corresponding plaintexts $P_3 = E^{-1}(C_3)$ and $P_4 = E^{-1}(C_4)$ through E^{-1}. Here the characteristic $\delta \longrightarrow \gamma$ holds twice for E_1^{-1} with probability one. As the boomerang condition suggests, we have $E_1^{-1}(C_3) \oplus E_1^{-1}(C_3) = \beta$.
- Check $P_3 \oplus P_4 = \alpha$. If this is true, identify the cipher as `FEAL`-8.

All probabilistic arguments work with probability one meaning that once the distinguisher given above works, the cipher can be identified as `FEAL`-8. Therefore, a 6-round trivial characteristic has been found for `FEAL`-8 and the differential attack mounted in previous chapter can be directly applied with this distinguisher. Now, the number of needed data for the attack is dramatically decreased. However, the

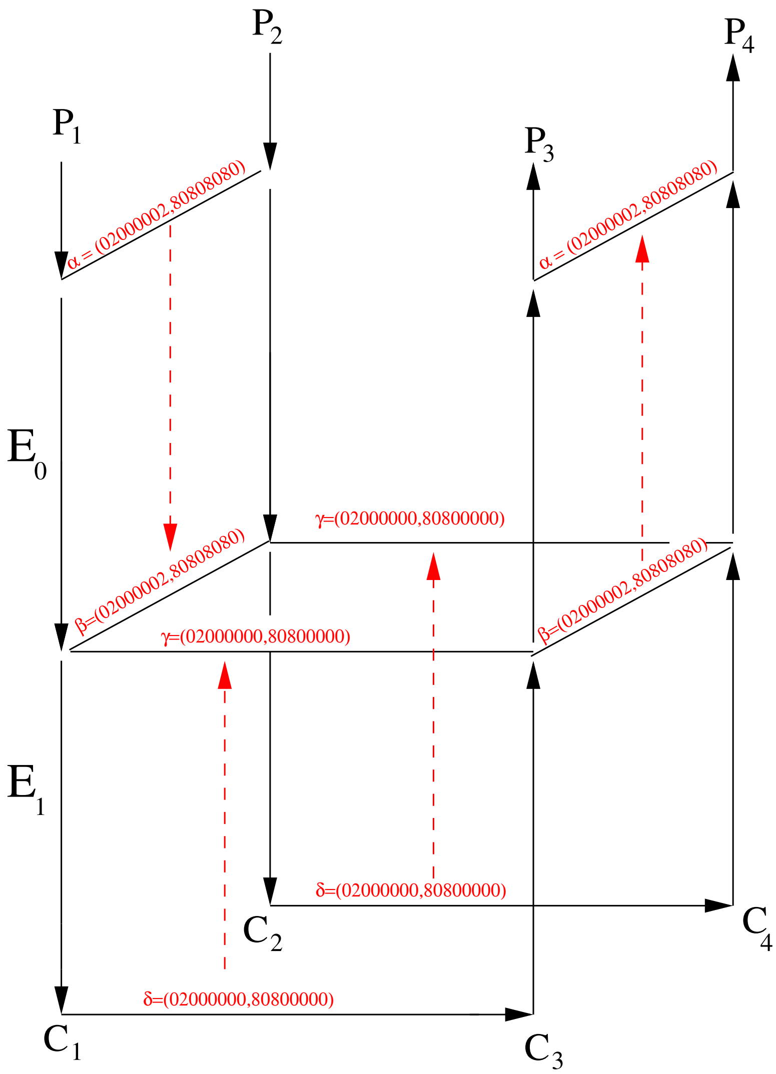


Figure 3.4: Boomerang Distinguisher on FEAL-8

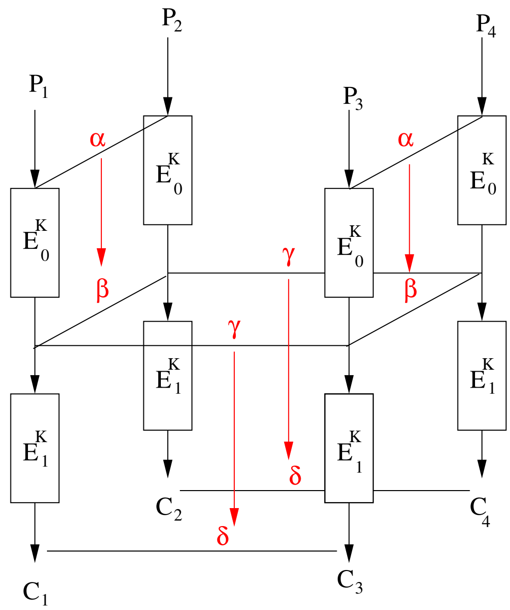


Figure 3.5: The Amplified Boomerang Distinguisher

attack is not a chosen plaintext attack, one needs 4 adaptively chosen plaintext and ciphertexts to apply boomerang attack on `FEAL-8`.

3.2 The Amplified-Boomerang and The Related-Key Amplified-Boomerang Attack

The amplified boomerang attack is the refinement of the boomerang attack introduced in previous section. The most important improvement in amplified boomerang attack is that it converts the adaptively chosen plaintext nature of boomerang attack into chosen plaintext attack. Instead of using both of the encryption and decryption boxes, amplified boomerang attack uses just the encryption box. However, probabilistically it is not a refinement as the convertion to the chosen plaintext scenario leads to some cost.

In amplified boomerang distinguisher, the attacker tries to construct the plaintext pairs (P_1, P_2) and (P_3, P_4) with a prescribed difference α. Here, through the encryption of these pairs, the differential $\alpha \to \beta$ holds with probability p for E_0. Assuming the independence of the plaintexts P_1 and P_3 (equivalently P_2 and P_4), at the end of E_0 the difference $E_0(P_1) \oplus E_0(P_3) = \gamma$ is expected to hold with probability 2^{-n}, given that the attacked cipher operates on n bits. Once this is satisfied, by the boomerang conditions introduced in previous section are satisfied and the difference $E_0(P_2) \oplus E_0(P_4) = \gamma$ ($E_0(P_1) \oplus E_0(P_3) = \gamma$ resp.) holds for free.

After constructing the boomerang condition at the end of E_0, for the second sub-cipher E_1 it is expected that the differential $\gamma \rightarrow \delta$ holds with probability q. Therefore, at the end of E_1, the differences $C_1 \oplus C_3 = \delta$ and $C_2 \oplus C_4 = \delta$ hold with probability $2^{-n}p^2q^2$. The amplified boomerang distinguisher is visualized in Figure 3.5 and can be summarized as:

- Take a randomly chosen plaintext P_1 and obtain the corresponding ciphertext $C_1 = E(P_1)$.
- Form $P_2 = P_1 \oplus \alpha$ and obtain the corresponding ciphertext $C_2 = E(P_2)$.
- Pick another randomly chosen plaintext P_3 and obtain the corresponding ciphertext $C_3 = E(P_3)$.
- Form $P_4 = P_3 \oplus \alpha$ and obtain the corresponding ciphertext $C_4 = E(P_4)$.
- Check $C_1 \oplus C_3 = \delta$ and $C_2 \oplus C_4 = \delta$.

Starting with N pairs $(P_1, P_2), (P_3, P_4)$ a fraction of about p satisfies the differential in E_0 (we expect Np pairs with output difference β in the input to E_1) and at the end of E_1 the expected number of right quartets satisfying the amplified boomerang distinguisher is $C(Np, 2).2^{-n}.q^2$, where $C(Np, 2)$ denotes the number of of ways choosing 2 pairs from Np pairs of plaintexts [28].

The related-key model of the amplified boomerang distinguisher is similar to the related-key model of the boomerang distinguisher and can be given as:

- Take a randomly chosen plaintext P_1 and obtain the corresponding ciphertext $C_1 = E_{K_1}(P_1)$.
- Form $P_2 = P_1 \oplus \alpha$ and obtain the corresponding ciphertext $C_2 = E_{K_2}(P_2)$, where $K_2 = K_1 \oplus \Delta K_{12}$.
- Pick another randomly chosen plaintext P_3 and obtain the corresponding ciphertext $C_3 = E_{K_3}(P_3)$, where $K_3 = K_1 \oplus \Delta K_{13}$.
- Form $P_4 = P_3 \oplus \alpha$ and obtain the corresponding ciphertext $C_4 = E_{K_4}(P_4)$, where $K_4 = K_3 \oplus \Delta K_{12}$.

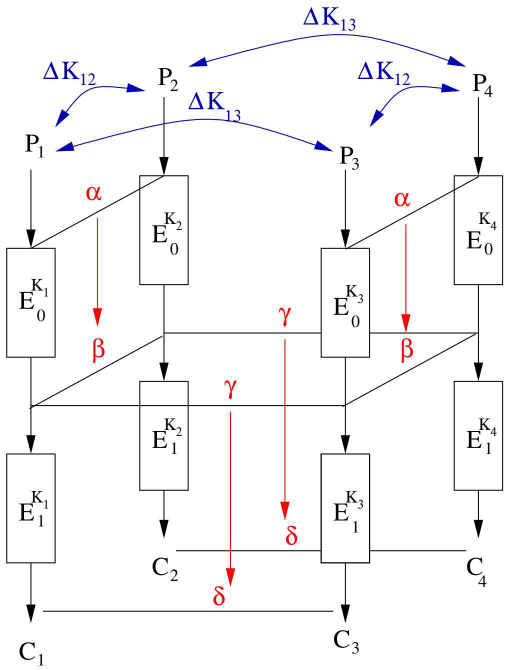


Figure 3.6: Related-Key Amplified Boomerang Distinguisher Based on Four Related Keys

- Check $C_1 \oplus C_3 = \delta$ and $C_2 \oplus C_4 = \delta$.

Next section covers the refinement of the amplified boomerang distinguisher, namely the rectangle distinguisher.

3.3 The Rectangle and The Related-Key Rectangle Attack

The rectangle attack also converts the adaptively chosen nature of the boomerang attack into the chosen plaintext attack. In fact, it is the refinement of the amplified-boomerang attack [37] and used to attack to many known ciphers [33, 43].

In boomerang distinguisher, the γ difference after E_0 and before E_1 is gathered through the decryption process. However, in rectangle distinguisher, the pairs (P_1, P_2) and (P_3, P_4) make use of the differential $\alpha \to \beta$ and since (P_1, P_3) is taken as random, it is expected that the difference $E_0(P_1) \oplus E_0(P_3) = \gamma$ holds with probability 2^{-n} before the subcipher E_1. Once this is satisfied, the differential $\gamma \to \delta$ comes to the picture. Of course, the subciphers before and after the rectangle distinguisher work as in the boomerang distinguisher.

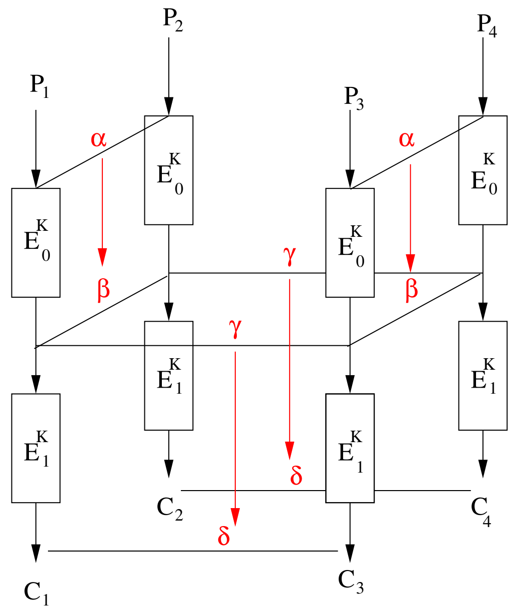


Figure 3.7: The Rectangle Distinguisher

Actually, so far, these are similar in amplified boomerang attack. Following [28], we will show the improvements specific to rectangle distinguisher. Firstly, besides the advantage of chosen plaintext nature, it also makes use of all β' values satisfying $\alpha \rightarrow \beta'$ and all γ' values that satisfy $\gamma' \rightarrow \delta$ as well. A right quartet $((P_1, P_2), (P_3, P_4))$ and corresponding ciphertexts $((C_1, C_2), (C_3, C_4))$ are defined such that

$$\begin{aligned} P_1 \oplus P_2 &= P_3 \oplus P_4 = \alpha \\ C_1 \oplus C_3 &= C_2 \oplus C_4 = \delta \end{aligned}$$

In the boomerang attack, it is known which ciphertext is derived from the other. In the rectangle attack this is not the case. For each pair (P_1, P_2) and (P_3, P_4), there are two possible quartets: $((P_1, P_2), (P_3, P_4))$ and $((P_1, P_2), (P_4, P_3))$. Each of these pairs can be tested to form a right quartet which reduces the data requirement. Using the notations given above, one can describe the related-key rectangle distinguisher based on 4 related-keys as follows (For further improvements, see [12, 28]).

- Take a randomly chosen plaintext P_1 and obtain the corresponding ciphertext $C_1 = E_{K_1}(P_1)$.

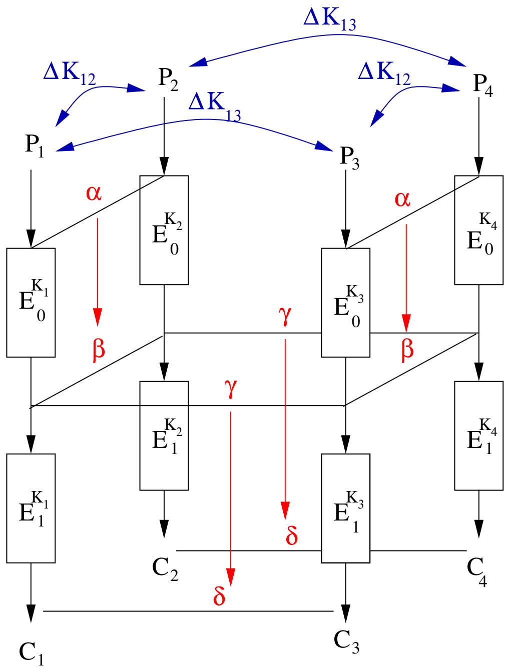


Figure 3.8: Related-Key Rectangle Distinguisher Based on Four Related Keys

- Form $P_2 = P_1 \oplus \alpha$ and obtain the corresponding ciphertext $C_2 = E_{K_2}(P_2)$, where $K_2 = K_1 \oplus \Delta K_{12}$.
- Pick another randomly chosen plaintext P_3 and obtain the corresponding ciphertext $C_3 = E_{K_3}(P_3)$, where $K_3 = K_1 \oplus \Delta K_{13}$.
- Form $P_4 = P_3 \oplus \alpha$ and obtain the corresponding ciphertext $C_4 = E_{K_4}(P_4)$, where $K_4 = K_3 \oplus \Delta K_{12}$.
- Check $C_1 \oplus C_3 = \delta$ and $C_2 \oplus C_4 = \delta$

The probability P of the rectangle distinguisher is given by $P = 2^{-n}\widehat{p}^2\widehat{q}^2$, where $\widehat{p} = \sqrt{\sum_\beta P^2_{K_1,K_2}(\alpha \rightarrow \beta)}$ and $\widehat{q} = \sqrt{\sum_\gamma P^2_{K_3,K_4}(\gamma \rightarrow \delta)}$. For a random cipher, the probability of the given difference is $P' = 2^{-2n}S$ where S is the cardinality of the set of differences of all δ values. Once $P \geq P'$ is satisfied, the rectangle distinguisher works. We conclude that given N plaintext pairs we expect to have $N^2\widehat{p}^2\widehat{q}^2/2^n$ right rectangle quartets.

As shown in [28], if the expected number of right quartet is taken to be 4, then there is at least one right quartet in the data set with probability 0.982 since it is a Poisson distribution with expectation of 4. Therefore, the number of plaintext pairs needed is $N = 2^{n/2+1}/\widehat{p}\widehat{q}$ that consist of $2^{n+2}/\widehat{p}^2\widehat{q}^2$ quartets expecting 4 right quartets at a time.

3.4 On The Security of The Encryption Modes of `MD4` and `MD5`

3.4.1 `MD4` and `MD5`

This subsection is devoted to well known hash functions `MD4` and `MD5` which inspire many of the popular hash functions as well such as `SHA`-family. `MD4` [69] was designed by Ron Rivest in 1990 which takes at most 2^{64} bits of input (as its padding rule permits) and produces 128-bit fingerprint of it. It is built on the Merkle-Damgärd Construction Principle that is proved to be a secure design principle [25]. The successor of `MD4`, `MD5` [70], was published again by Ron Rivest two years later by a slight modifications on the compression function.

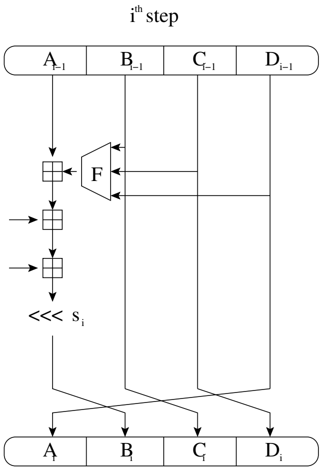
`MD4` takes message blocks of 512 bits and produces 128-bit hash value. In order to make the message an exact multiple of 512-bit, the padding procedure is applied. First, one 1 bit and enough 0 bits are added to the end of the message to make the length 448 modulo 512. Finally, 64-bit representation of the original message length is filled for the remaining 64 bits. After padding, message becomes exact multiple of 512 bits, so is an exact multiple of 16 (32-bit) words.

`MD4` has 3 rounds, each consisting 16 steps. In every step i, the state variables $(A_{i-1}, B_{i-1}, C_{i-1}, D_{i-1})$ are updated as A_i, B_i, C_i, D_i by using the message word M_i, the step constant K_i and the shift value s_i specified for the step i. A_{i-1} , B_{i-1} , C_{i-1} and D_{i-1} are intermediate variables which are updated at every step. The Figure 3.9(a) visualizes one step of the step function of `MD4`.

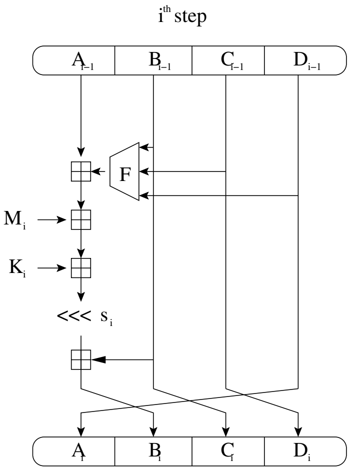
Every round of `MD4` compression function uses a different boolean function as stated below:

$$\begin{aligned} F_1(X,Y,Z) &= XY \oplus XZ \oplus Z \\ F_2(X,Y,Z) &= XY \oplus XZ \oplus YZ \\ F_3(X,Y,Z) &= X \oplus Y \oplus Z \end{aligned}$$

`MD4` type hash functions use a public IV as an initial state variable. Exact values of IV, permutation of words, constants and shift values are given in [69].



(a) MD4



(b) MD5

Figure 3.9: MD4 and MD5

In 1992, Ron Rivest made some refinements on MD4 algorithm and published the new version as MD5. These refinements include the addition of the fourth round as $F_4(X, Y, Z) = Y \oplus (X \oplus \neg Z)$ (where $\neg$ denotes the bitwise complement of the state variable) and the change of the second round (that is, F_2 is $F_2(X, Y, Z) = XY \oplus Y \oplus Z \oplus 1$). Besides, the state variable B_i is added to the output of the boolean function at each step (The full list of changes including the constants and the shift values are given in [70]). The Figure 3.9(b) visualizes one step of the compression function of MD5.

3.4.2 Related-Key Boomerang and Rectangle Attacks to the Encryption Mode of MD4

In this and the following subsection, the related-key boomerang and rectangle attacks of Kim *et al* [42] to the encryption mode of MD4 based on 4-related-keys are presented. In their work, there exist also other attacks based on 2-related keys and some weak classes.

Before starting the attack procedure, the encryption mode of MD4 has to be identified which can be described easily. The encryption mode of MD4 is supposed to encrypt 128-bit plaintext to 128-bit ciphertext using 512-bit secret key. The decryption is also well-defined as there is no need to inverse the round operations. The message expansion (key scheduling) of MD4 is a permutation of 16 32-bit words that are used used exactly once in each round pass in a specified order. In this attack, the attackers make use of the linearity in message expansion algorithm.

To construct an efficient differential characteristic for the message expansion of MD4, the behavior of the message words in the subciphers E_0 and E_1 should be investigated carefully. First of all, all the message differences introduced in E_0 and E_1 are constructed by just flipping the most significant bit of the message words (the difference denoted by $\Delta M_i = e_{31}$) as it kills the carry effect in modular addition. Then, the message words to be changed are chosen such that two appearances of them are as wide as possible to construct a long distinguisher.

For MD4, the rounds between $0-29$ and $29-47$ are chosen as E_0 and E_1 respectively. The message word to be changed in E_0 is M_3 and the message word to be changed in E_1 is chosen to be M_7. These message words are determined such that the distinguisher covers all the rounds in MD4. Table 3.1 shows the characteristic used for MD4. We use the notation given in [42]. Here e_i denotes the difference that has zero difference in all bits except the i^{th} bit. Similarly, $e_{i_1,i_2,..,i_k}$ denotes the difference $e_{i_1} \oplus e_{i_2} \oplus ... \oplus e_{i_k}$. In the table, $Pr[REC]$ and $Pr[BOO]$ denote the probability of the related-key rectangle and boomerang distinguishers respectively. $Pr[REC]$ is given by

$$Pr[REC] = (2^{-2})^2 1^2 2^{-128} = 2^{-132}.$$

Since the differences after step 45 can be seen in the ciphertext directly, the probabilities after step 45 are discarded. Similarly, $Pr[BOO]$ is given by

$$Pr[BOO] = 2^{-2}(2^{-1})^2 1 = 2^{-4}.$$

The first factor is p and the second factor is q^2. However, for the last factor the first 3 steps are discarded as the differences are expected in the plaintexts.

The related-key boomerang attack on MD4 can be described as follows:

- Prepare 2^5 plaintext pairs (P_1, P_2) with $\Delta = (0, e_{31}, 0, 0)$. By the properties of the boolean function $F_1 = XY \oplus XZ \oplus Z$, the most significant bits of C and D have to be equal.
- Obtain the corresponding ciphertexts $C_1 = E_{m_1}(P_1)$ and $C_2 = E_{m_2}(P_2)$ through E, where $m_2 = m_1 \oplus (0, 0, 0, e_{31}, 0, ..., 0)$.
- Calculate the second ciphertext pair as $C_3 = C_1 \oplus \delta$ and $C_4 = C_2 \oplus \delta$.
- Obtain the corresponding 2^5 plaintexts $P_3 = E^{-1}_{m_3}(C_3)$ and $P_4 = E^{-1}_{m_4}(C_4)$

Round	ΔA	ΔB	ΔC	ΔC	ΔK	$Probability$
α	0	e_{31}	0	0	K_{12}	
0	0	e_{31}	0	0	0	1
1	0	0	e_{31}	0	0	2^{-1}
2	0	0	0	e_{31}	0	2^{-1}
3	e_{31}	0	0	0	e_{31} (ΔM^3)	1
4	0	0	0	0	0	1
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
27	0	0	0	0	0	1
28	0	0	0	0	e_{31} (ΔM^3)	1
β	0	e_2	0	0		$p = 2^{-2}$
γ	e_{31}	0	0	0	K_{13}	
29	e_{31}	0	0	0	e_{31} (ΔM^7)	1
30	0	0	0	0	0	1
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
45	0	0	0	0	0	1
46	0	0	0	0	e_{31} (ΔM^7)	1
47	0	e_{10}	0	0	0	2^{-1}
δ	0	e_{25}	e_{10}	0		$q = 2^{-1}$
REC						$Pr[REC] \approx 2^{-132}$
BOO						$Pr[BOO] \approx 2^{-4}$

Table 3.1: The characteristic for `MD4`

through E^{-1}, where $m_3 = m_1 \oplus (0,0,0,0,0,0,0,e_{31},0,...,0)$,
$m_4 = m_3 \oplus (0,0,0,0,0,0,0,e_{31},0,...,0)$.

- Check $P_3 \oplus P_4 = P_1 \oplus P_2$.

- If this is the case, identify the corresponding cipher as MD4.

Since we gathered 2^5 plaintext pairs and $Pr[BOO] = 2^{-4}$, the probability that a pair is not a right quartet is $(1 - 2^{-4})$. So, the probability that the attack succeeds is $1 - (1 - 2^{-4})^{2^5} = 0.87$. For the related-key rectangle distinguisher on the other hand, the attack can be described as follows:

- Prepare 2^{67} plaintext pairs (which construct 2^{134} quartets) (P_i, P_i^*) with $\Delta = (0, e_{31}, 0, 0)$. By the properties of the boolean function F_1, the most significant bits of C and D have to be equal.

- Obtain the corresponding ciphertexts $C_1 = E_{m_1}(P_1)$ and $C_2 = E_{m_2}(P_2)$ through E, where $m_2 = m_1 \oplus (0,0,0,e_{31},0,...,0)$ and obtain $C_3 = E_{m_3}(P_3)$ and $C_4 = E_{m_4}(P_4)$ through E, where $m_4 = m_3 \oplus (0,0,0,e_{31},0,...,0)$.

- Check if there exist $C_1 \oplus C_3 = C_2 \oplus C_4$.

- If this is the case, identify the corresponding cipher as MD4.

Since we gathered 2^{67} plaintext pairs and $Pr[REC] = 2^{-132}$, the probability that a pair is not a right quartet is $(1 - 2^{-132})$. So, the probability that the attack succeeds is $1 - (1 - 2^{-132})^{2^{134}} = 0.98$.

3.4.3 Related-Key Boomerang and Rectangle Attacks to the Encryption Mode of MD5

The Related-Key Boomerang and Rectangle Attacks to the encryption mode of MD5 is similar to the attack presented in previous subsection. Here, we will just give the characteristic used for MD5. Table 3.2 summarizes the attack where $Pr[REC] \approx 2^{-137.1}$ and $Pr[BOO] \approx 2^{-11.6}$.

Round	ΔA	ΔB	ΔC	ΔC	ΔK	$Probability$
α	0	0	e_{31}	0	K_{12}	
0	0	0	e_{31}	0	0	1
1	0	0	0	e_{31}	0	2^{-1}
2	e_{31}	0	0	0	e_{31} (ΔM^2)	1
3	0	0	0	0	0	1
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
28	0	0	0	0	0	1
29	0	0	0	0	e_{31} (ΔM^2)	2^{-1}
30	0	e_8	0	0	0	$p = 2^{-2}$
β	0	e_8	e_8	0		$p = 2^{-4}$
γ	$e_{11,31}$	e_{31}	e_{31}	e_{31}	K_{13}	
31	$e_{11,31}$	e_{31}	e_{31}	e_{31}	0	2^{-1}
32	0	0	e_{31}	e_{31}	0	1
33	e_{31}	0	0	e_{31}	0	1
34	e_{31}	0	0	0	e_{31} (ΔM^{11})	1
35	0	0	0	0	0	1
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
60	0	0	0	0	0	1
61	0	0	0	0	e_{31} (ΔM^{11})	2^{-1}
62	0	e_9	0	0	0	2^{-2}
63	0	e_9	e_9	0	0	2^{-2}
δ	0	e_9	e_9	e_9		$q = 2^{-6}$
REC						$Pr[REC] \approx 2^{-137.1}$
BOO						$Pr[BOO] \approx 2^{-11.6}$

Table 3.2: The characteristic for `MD5`

Chapter 4

Previous Collision Attacks on the `Tiger` Hash Function

Recent developments in cryptanalysis of dedicated hash functions brought new attention to the cryptanalysis of alternative hash functions, such as `Tiger` [2]. `Tiger` [2] is an important type of an hash function which is proved to be secure so far as there is no known collision attack on the full `Tiger`. It is designed by Biham and Anderson in 1995 to be very fast on modern computers, and in particular on the 64-bit computers, while it is still not slower than other suggested hash functions on 32-bit machines.

Recently some weaknesses have been found for `Tiger`-hash function. First, in FSE ′06 [38] Kelsey and Lucks found a collision for 16-17 rounds of `Tiger` and a pseudo-near-collision for 20 rounds. Then, in INDOCRYPT ′06, Mendel *et al* [55] extended this attack to 19-round collision and 22-round pseudo-near-collision. Finally in 2007 at ASIACRYPT ′07 Mendel and Rijmen [56] found a pseudo-near-collision for the full `Tiger`. Therefore, the cryptanalysis of the `Tiger`-hash function is a popular research area in these days.

This chapter is devoted to the some of the previous collision search attacks to `Tiger` hash function. First, the basic attack strategy is introduced which is the common feature of all the attacks presented. Then, the attack by Kelsey and Lucks and the extension of this attack by Mendel *et al* will be mentioned. The former is the first attempt to find collisions for `Tiger` and the latter is the first attack mounted on full `Tiger`. Table 5.1 summarizes the collision search attacks to the `Tiger` hash function (an extension of the table in [55]).

Number of Rounds	Attack Type	Attack	Complexity
16	Collision	[38]	2^{44}
19	Collision	[55]	2^{62} and 2^{69}
19	Pseudo-Collision	[55]	2^{44}
21	Pseudo-Collision	[55]	2^{66}
20	Pseudo-Near-Collision	[38]	2^{48}
21	Pseudo-Near-Collision	[55]	2^{44}
22	Pseudo-Near-Collision	[55]	2^{44}
23	Pseudo-Near-Collision	[56]	2^{47}
24	Pseudo-Near-Collision	[56]	2^{47}

Table 4.1: Overview of Attacks to the `Tiger` Hash Function

Notation	Meaning
$A \boxplus B$	Addition of A and B mod 2^{64}
$A \boxminus B$	Subtraction of B from A mod 2^{64}
$A \boxtimes B$	Multiplication of A and B mod 2^{64}
$A \oplus B$	Bitwise XOR-operation of A and B mod 2^{64}
$\neg A$	Bitwise NOT-operation of A
$A << n$	Bitwise rotation of A to the left by n bits
$A >> n$	Bitwise rotation of A to the right by n bits
X_i	Message word i
$X_i[\mathbf{even}]$	Even bytes of X_i
$X_i[\mathbf{odd}]$	Odd bytes of X_i
M^i	Message i
X_j^i	Message word j of the Message i
A_j^i	State Variable j of the Message i

Table 4.2: Notation

4.1 `Tiger`

`Tiger` [2] is a cryptographic, iterative hash function which is designed for 64-bit processors by Biham and Anderson in 1995. Its compression function is based on a block-cipher-like function producing 192-bit hash value from a 512-bit message block. The size of the hash value and the intermediate state length are 192-bit (three 64-bit words). A detailed description of `Tiger` is given in [2] and for the remaining parts, we will follow the notation given in Table 4.2.

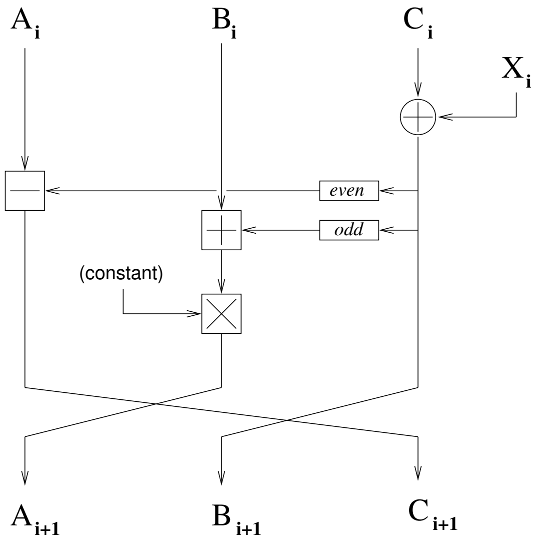


Figure 4.1: The i^{th} Round of `Tiger`

4.1.1 The Round Function of `Tiger`

In `Tiger`, state update transformation starts with a fixed `IV` and at each round three 64-bit state variables A, B, C are updated as follows:

$$
\begin{aligned}
A &:= A \boxminus even(C) \\
B &:= (B \boxplus odd(C)) \ \boxtimes \ mult \text{ , where } mult \in \{5,7,9\} \\
C &:= C \oplus X_i
\end{aligned}
$$

Here, each 64-bit message word is obtained from 512-bit message block and the non-linear functions *even* and *odd* operate on even and odd bytes of the input respectively that are defined as:

$$
\begin{aligned}
even(C) &: = t_1(C[0]) \oplus t_2(C[2]) \oplus t_3(C[4]) \oplus t_4(C[6]) \\
odd(C) &: = t_1(C[7]) \oplus t_2(C[5]) \oplus t_3(C[3]) \oplus t_4(C[1])
\end{aligned}
$$

Four 8×64 bit S-boxes are used in even and odd functions and shown by t_1, t_2, t_3 and t_4 where $C[i]$ denotes the i^{th} byte of C ($0 \leq i \leq 7$, 7^{th} byte is the most significant byte). The i^{th} round input values are shown by A_i, B_i, C_i (three 64-bit words) where $i \in \{0, \ldots, 24\}$, i^{th} round message block is X_i and i^{th} round output values are A_{i+1}, B_{i+1}, C_{i+1}. One step of `Tiger` is shown in Figure 4.1

Before the beginning of the second 8-round pass, intermediate values A, B, C

are updated as C_8, A_8, B_8. Before the beginning of the last 8-round pass again intermediate values are updated and they are assigned to B_{16}, C_{16}, $A_{16}[2]$ (As there is no effect of this simple operation, all attacks including ours discard this operation). After the last round of the state update transformation, the initial value A_0, B_0, C_0 and the output of the last round A_{24}, B_{24}, C_{24} are combined resulting to the hash value or the initial value of the next step as follows.

$$\begin{aligned} A_{25} &= A_0 \oplus A_{24} \\ B_{25} &= B_0 \boxminus B_{24} \\ C_{25} &= C_0 \boxplus C_{24} \end{aligned}$$

The block cipher mode of `Tiger` is straightforward. The chaining operations of the intermediate values are omitted and `Tiger` is treated as a block cipher encrypting 192-bit plaintext into 192-bit ciphertext using 512-bit secret key. There is no need to invert *odd* and *even* function since their inverses do not affect the decryption mode. In the decryption mode, the inverses of the modular operations are used which can be defined very easily.

4.1.2 The Message Expansion of `Tiger`

In `Tiger`, 512-bit message block is expanded to 24×64-bit message blocks by using a message expansion algorithm. The non-linear invertible message expansion of `Tiger` uses some logical operations. In the first 8 rounds, the original message words $X_0, ..., X_7$ are used and for the next 8 rounds the message expansion is applied to $X_0, ..., X_7$ and the message words $X_8, ..., X_{15}$ are formed. For the remaining 8 rounds the message expansion is performed to the message words $X_8, ..., X_{15}$ to gather $X_{16}, ..., X_{23}$. 512-bit key is expanded by using the operations shown in Table 4.3.

4.2 Attack Method

The basic attack strategy for the collision search attacks to the `Tiger` hash function basically consists of the method developed by Kelsey and Lucks in [38]. The extensions of this attack by Mendel *et al* in [55] and in [56] use the same attack strategy but with some slight modifications.

First Pass	Second Pass
$X_0 = X_0 \boxminus (X_7 \oplus A5A5A5A5A5A5A5A5_x)$	$X_0 = X_0 \boxplus X_7$
$X_1 = X_1 \oplus X_0$	$X_1 = X_1 \boxminus (X_0 \oplus (\overline{X_7} << 19))$
$X_2 = X_2 \boxplus X_1$	$X_2 = X_2 \oplus X_1$
$X_3 = X_3 \boxminus (X_2 \oplus (\overline{X_1} << 19))$	$X_3 = X_3 \boxplus X_2$
$X_4 = X_4 \oplus X_3$	$X_4 = X_4 \boxminus (X_3 \oplus (\overline{X_2} >> 23))$
$X_5 = X_5 \boxplus X_4$	$X_5 = X_5 \oplus X_4$
$X_6 = X_6 \boxminus (X_5 \oplus (\overline{X_4} >> 23))$	$X_6 = X_6 \boxplus X_5$
$X_7 = X_7 \oplus X_6$	$X_7 = X_7 \boxminus (X_6 \oplus 0123456789ABCDEF_x)$

Table 4.3: The Message Expansion of `Tiger`

The attack starts with finding a good differential characteristic for the message expansion of `Tiger`. This part generally works with probability one. The second part of the algorithm makes use of the message modification technique for `Tiger` hash function. In this part, some necessary differences are introduced to the state variables. Then, these differences are cancelled by the differences introduced by the message expansion algorithm. Before the introduction of these two parts, some notation and conventions will be given as they are used in all attacks mounted on `Tiger`.

4.2.1 Conventions

In this and the following chapters, we follow the notation and conventions introduced by Kelsey and Lucks [38]. First, throughout the attack, in order to avoid misunderstandings we will make a difference between the additive differences and the XOR difference. We will use the following notation for the additive and the XOR differences.

$$\begin{aligned} \Delta^{+}(X) &= X \boxminus X^{*} \text{ , additive difference mod } 2^{64} \\ \Delta^{\oplus}(X) &= X \oplus X^{*}\text{, XOR difference} \end{aligned}$$

The differences between the message words are seen as the XOR difference as it is XORed to the state variables. However, the differences in the state variables are seen as the additive differences as the addition and the subtraction are used as the basic operations in the compression function of `Tiger`.

Moreover, we will switch between the additive differences to the XOR difference or vice verse. Generally, it works with probability 1/2, except the most significant bit of the words. Namely,

$$\begin{aligned} \text{If } X \boxminus X^* &= 2^i, \text{ then, the probability } P[X \oplus X^* = 2^i] = 1/2. \\ \text{The exception is } i &= 63, \text{where } P[X \oplus X^* = 2^i] = 1. \end{aligned}$$

The attacks proposed for `Tiger` make an extensive use of this fact. Let I denote the difference of 2^{63}. Then, there exist no difference between the additive differences and the XOR difference probabilistically. Furthermore, during the attacks, the attackers use the another simple trivial fact that the I difference propagate as difference I through *multiplication* and zero difference through *even* function. The simple proof of the former statement is given below.

CLAIM : For an odd constant c, $c.2^{63} \equiv 2^{63}(mod 2^{64})$(that is, $c.I \equiv I(mod 2^{64})$).

Proof : Let us assume the contrary. That is, $2^{64} \nmid c.2^{63} - 2^{63}$. Say $c = 2t + 1$. Then, $2^{64} \nmid (2t + 1).2^{63} - 2^{63} \;\Rightarrow\; 2^{64} \nmid 2^{64}t$ which is a contradiction. Therefore, the difference I propagates as I through the multiplication by an odd constant.

Following these tricks, the collision search attacks are going to be presented in the following sections.

4.2.2 Finding Good Differential Characteristics For The Message Expansion of `Tiger`

In `Tiger`, the message expansion algorithm is non-linear. However, some differences propagate linearly. Some of such differentials are used in [38, 55, 56] to attack `Tiger`. In [38] Kelsey and Lucks are assumed to find the used characteristic exhaustively by imposing all possibilities of the distribution of the I difference in the message words. In [55, 56] Mendel *et al*, on the other hand, used a linearized model of the key schedule by imposing coding theory and applying some efficient probabilistic algorithms (The details are given in [56, 65]).

This motivates us to search for other good differentials that propagate very efficiently. What makes it good in terms of their efficiency is quite obvious in that the hamming weight of the corresponding differences should be kept small. Also, reducing carry effect by introducing the difference I, we got several probability one differentials, some of them are given in Table 4.4. The table consists of the characteristics where in all rounds the I difference appears. These differences are used to cancel the differences imposed on the state variables which will be mentioned in the

The Propagation of Differences			
Message Differences	Rounds $0-7$	Rounds $8-15$	Rounds $16-23$
$(0,0,0,0,I,I,I,I)$	$(0,0,0,0,I,I,I,I)$	$(0,I,0,I,I,0,0,I)$	$(0,0,0,I,I,I,I,0)$
$(0,0,0,I,0,0,0,I)$	$(0,0,0,I,0,0,0,I)$	$(0,I,0,0,0,0,0,I)$	$(0,0,0,0,0,0,0,I)$
$(0,0,0,I,I,I,I,0)$	$(0,0,0,I,I,I,I,0)$	$(0,0,0,I,I,0,0,0)$	$(0,0,0,I,I,I,I,I)$
$(0,0,I,0,0,0,I,I)$	$(0,0,I,0,0,0,I,I)$	$(I,0,0,0,0,0,I,I)$	$(0,0,0,0,0,0,I,I)$
$(0,0,I,0,I,I,0,0)$	$(0,0,I,0,I,I,0,0)$	$(I,I,0,I,I,0,I,0)$	$(0,0,0,I,I,I,0,I)$
$(0,0,I,I,0,0,I,0)$	$(0,0,I,I,0,0,I,0)$	$(I,I,0,0,0,0,I,0)$	$(0,0,0,0,0,0,I,0)$
$(0,0,I,I,I,I,0,I)$	$(0,0,I,I,I,I,0,I)$	$(I,0,0,I,I,0,I,I)$	$(0,0,0,I,I,I,0,0)$
$(0,I,0,0,0,I,I,I)$	$(0,I,0,0,0,I,I,I)$	$(0,0,0,0,0,I,I,0)$	$(0,0,0,0,0,I,I,I)$
$(0,I,0,0,I,0,0,0)$	$(0,I,0,0,I,0,0,0)$	$(0,I,0,I,I,I,I,I)$	$(0,0,0,I,I,0,0,I)$
$(0,I,0,I,0,I,I,0)$	$(0,I,0,I,0,I,I,0)$	$(0,I,0,0,0,I,I,I)$	$(0,0,0,0,0,I,I,0)$
$(0,I,0,I,I,0,0,I)$	$(0,I,0,I,I,0,0,I)$	$(0,0,0,I,I,I,I,0)$	$(0,0,0,I,I,0,0,0)$
$(0,I,I,0,0,I,0,0)$	$(0,I,I,0,0,I,0,0)$	$(I,0,0,0,0,I,0,I)$	$(0,0,0,0,0,I,0,0)$
$(0,I,I,0,I,0,I,I)$	$(0,I,I,0,I,0,I,I)$	$(I,I,0,I,I,I,0,0)$	$(0,0,0,I,I,0,I,0)$
$(0,I,I,I,0,I,0,I)$	$(0,I,I,I,0,I,0,I)$	$(I,I,0,0,0,I,0,0)$	$(0,0,0,0,0,I,0,I)$
$(0,I,I,I,I,0,I,0)$	$(0,I,I,I,I,0,I,0)$	$(I,0,0,I,I,I,0,I)$	$(0,0,0,I,I,0,I,I)$

Table 4.4: The Propagation of Some Differences with probability 1

following sections in detail.

4.2.3 Message Modification by Meeting in the Middle

In the first attack [38], to find collision in reduced round `Tiger`, Kelsey and Lucks proposed a new type of message modification technique modified for `Tiger` by meeting in the middle. The key primitive of this technique is the use of the degree of freedom in the choice of message words to impose conditions on the state variables. Thus, one can impose some differences to state variables by message modification and then these differences are cancelled with the differences coming from the message expansion algorithm.

Assume we are given the state variables A_i, B_i, C_i (A_i^*, B_i^*, C_i^*) and $X_i[even]$ together with the message differences $\Delta^{\oplus}(X_i)$ and $\Delta^{\oplus}(X_{i+1})$. Automatically we have the modular differences $\Delta^{+}(A_i), \Delta^{+}(B_i)$ and $\Delta^{+}(C_i)$. Then, the modular difference $\Delta^{+}(C_{i+2})$ can be forced to be any modular difference δ with probability $1/2$ by using the birthday attack. We will briefly describe the method in Figure 4.2.

As shown in the figure, the additive difference δ depends on additive differences shown by the differences on the dashed-lines. From now on, we need to consider the additive differences and the XOR differences at the same time because the reason why we apply the message modification is to fix the message words X_i and X_{i+1}

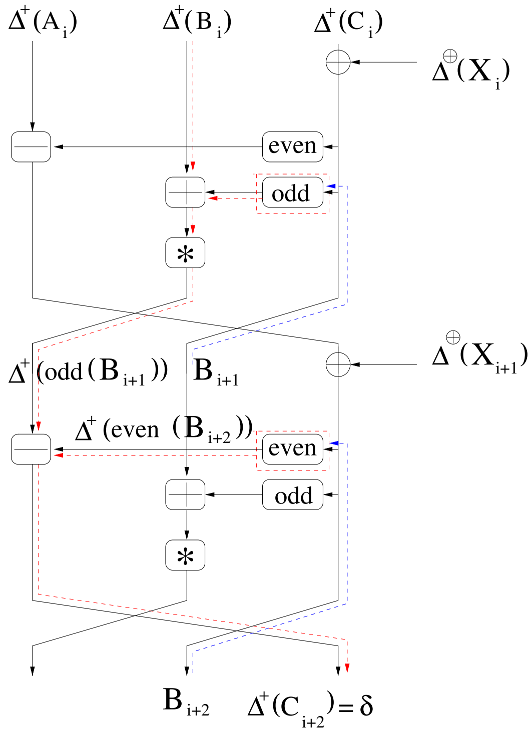


Figure 4.2: Message Modification by Meet-in-the-Middle

and they are added to the state variables by the XOR operation. For any nonzero XOR difference $\Delta^{\oplus}(B_{i+2}[even])$, we expect to have about 2^{32} different corresponding additive output differences $\Delta^{+}(even(B_{i+2}))$ and for any nonzero XOR difference $\Delta^{\oplus}(B_{i+1}[odd])$ (Here, in our notation $\Delta^{\oplus}(B_{i+1}[odd])$ contains $\Delta^{+}(B_i)$), we expect to have about 2^{32} different corresponding additive output differences $\Delta^{+}(odd(B_{i+1}))$. Note that these differences should be taken as nonzero additive or XOR differences.

The meet-in-the-middle approach works as follows.

1. Store the 2^{32} candidates of $\Delta^{+}(odd(B_{i+1}))$ in a table by guessing $X_i[odd]$

$$
\begin{aligned}
odd(B_{i+1}) &= (B_i \boxplus odd(C_i[odd] \oplus X_i[odd])) \boxtimes mult \\
odd(B^*_{i+1}) &= (B^*_i \boxplus odd(C^*_i[odd] \oplus X^*_i[odd])) \boxtimes mult
\end{aligned}
$$

   Note that B_i, B^*_i, C_i, C^*_i and $\Delta^{\oplus}(X_i)$ are known.

2. For all 2^{32} candidates of $\Delta^{+}(even(B_{i+2}))$, calculate $\Delta^{+}(even(B_{i+2})$ by guessing $X_{i+1}[even]$

$$
even(B_{i+2}) = even(A_i \boxminus even(C_i[even] \oplus X_i[even]) \oplus X_{i+1}[even]
$$

$$even(B^*_{i+2}) \quad = \quad even(A^*_i \boxminus even(C^*_i[even] \oplus X^*_i[even]) \oplus X^*_{i+1}[even]$$

Note that $A_i, A^*_i, C_i, C^*_i, X_i[even]$ and $\Delta^{\oplus}(X_{i+1}))$ are known.

3. Test whether there exist some $\Delta^+(odd(B_{i+1}))$ satisfying $\Delta^+(odd(B_{i+1})) = (\Delta^+(even(B_{i+2})) \boxplus \delta)$.

This technique needs about 2^{33} evaluations round function of `Tiger` that is equivalent to about $2^{28.5}$ evaluations of the `Tiger` compression function. Data complexity of the precomputation is 2^{32} units (each unit is 2^3-byte) of storage space. In the attack scenario, we assumed that the message word $X_i[even]$ has been fixed and the meet-in-the-middle approach gathered 64 local message bits $X_i[odd]$ and $X_{i+1}[even]$. Therefore, at the end of this step we completed the message word X_i and we gathered $X_{i+1}[even]$.

4.3 Collision Attack on `Tiger`-16

In the collision attack on `Tiger`-16 [38], Kelsey and Lucks first found a differential characteristic in message expansion working with probability one. Then, by using the message modification technique introduced in the previous chapter, they imposed differences on state variables to cancel the differences coming from the message expansion algorithm. The reason why they attacked to reduced round `Tiger`-16 is quite obvious in the sense that they used an effective differential characteristic in message expansion algorithm which is very suitable for attacking 16 rounds as canceling after round 10 leads to a collision at the end of the round 16. The attack can be broken into three pieces [38]:

1. Use the differential characteristic $(I, I, I, I, 0, 0, 0, 0) \longrightarrow (I, I, 0, 0, 0, 0, 0, 0)$ in the message expansion algorithm (Note that the path does not consider the last 8 rounds).
2. Make use of the differential characteristic $(I, I, 0) \longrightarrow (0, 0, 0)$ in rounds $7-10$ of the round function. (Since the message words in rounds $10-15$ are unchanged, this leads to a collision after 16 rounds.)
3. Use message modification to force the difference in the round function after round 6 to $(I, I, 0)$.

In order to have a collision in the compression function of `Tiger` after 16 rounds, it is required that there is a collision after round 9. Hence, the following differences are needed in the state variables for round 7 of `Tiger`.

$$\Delta^{\oplus}(A_7) = I, \Delta^{\oplus}(B_7) = I, \Delta^{\oplus}(C_7) = 0$$

The most important part of the attack is to construct the desired difference in the state variables for round 7. Kelsey and Lucks use the message modification technique described in previous section. The following subsections contain the necessary modifications to construct the desired difference.

i	ΔA_i	ΔB_i	ΔC_i	ΔX_i
0	0	0	0	I
1	*	*	*	I
2	*	*	*	I
3	*	*	*	I
4	*	*	$K^{\oplus}$	I
5	0	K^{+}	$L^{\oplus}$	0
6	0	L^{+}	I	0
7	I	I	0	0
8	I	0	I	I
9	0	0	I	I
10	0	0	0	0
11	0	0	0	0
12	0	0	0	0
13	0	0	0	0
14	0	0	0	0
15	0	0	0	0

Table 4.5: The Characteristic for Collision Attack on `Tiger`-16

4.3.1 Precomputation

The precomputation step is used for all attacks performed on `Tiger`. We will use the notation given in [56]. The reason why we applied precomputation step is that we have to find a set L of possible modular differences L^{+} which are consistent to a low weight XOR-difference $L^{\oplus}$. It is said that a modular difference L^{+} is consistent to $L^{\oplus}$ if there exist X and X such that $X^{*} \oplus X = L^{\oplus}$ and $X^{*} \boxminus X = L^{+}$. An example of a consistency between XOR and modular differences are given in table 4.6.

Let $C_{L^{\oplus}}$ be the set of modular differences L^{+} which are consistent to the XOR-difference $L^{\oplus}$. Then the set L of modular differences for the collision attack is defined

x_i	$\Delta^{\oplus} = 0100$	$x_i \oplus 0100 = x_i^*$	$x_i \boxminus x_i^* = L^+$
0000	0100	0100	1100
0001	0100	0101	1100
0010	0100	0110	1100
0011	0100	0111	1100
0100	0100	0000	0100
0101	0100	0001	0100
0110	0100	0010	0100
0111	0100	0011	0100
1000	0100	1100	1100
1001	0100	1101	1100
1010	0100	1110	1100
1011	0100	1111	1100
1100	0100	1000	0100
1101	0100	1001	0100
1110	0100	1010	0100
1111	0100	1011	0100
	$h_w = 1$		$\|L^+\| = 2$

Table 4.6: An Example of Consistency Between XOR and Modular Difference

as:

$$L = \{L^+ \in C_{L^\oplus} : L^+ = I \boxplus odd(B_7 \oplus I) \boxminus odd(B_7)\}$$

The cardinality of the set $C_{L^\oplus}$ is directly related to the Hamming weight of $L^\oplus$, namely $|C_{L^\oplus}| \leq 2^{HW(L^\oplus)}$ (An n-bit XOR difference has either 2^n or 2^{n-1} additive differences consistent with it, depending on the bit position flipped). Obviously, the differences with low Hamming weight are used for $L^\oplus$. Kelsey and Lucks claimed that there exists $L^\oplus$ of Hamming weight 8 and this value can be used in their attack. However, Mendel and Rijmen [56] claimed that in their modified collision attack on `Tiger`-16 that the Hamming Weight of 10 can be found for $L^\oplus$. However, what is important about the attack is the number of elements in L.

Similarly, a set K of possible modular differences K^+ which are consistent to the XOR-difference $K^\oplus$ can be constructed:

$$K \quad = \quad \{K^+ \in C_{K^\oplus} : K^+ = odd(B_6 \oplus L^\oplus) \boxminus odd(B_6)\}.$$

Here, we applied the attack with the exact values of $L^\oplus$ and $K^\oplus$. The XOR differences $L^\oplus$ and $K^\oplus$ of Hamming Weight 12 and 8 are used respectively. Obviously, it is impossible to search for all differences of the prescribed Hamming weight because of the time complexity. Instead, we start searching for the differences satisfying the below relation,

$$L = \{L^+ \in C_{L^\oplus} : L^+ = I \boxplus odd(B_7 \oplus I) \boxminus odd(B_7)\}$$

All 2^{32} possibilities of the state variable B_7 were tried and the corresponding differences were gathered. Here, we could not find the modular differences consistent with an XOR difference of Hamming Weight $8, 9$ and 10. Therefore, we extended the search for higher Hamming Weights of 11 and 12. Four XOR differences of Hamming weight 11 and 61 XOR differences of Hamming weight 12 are found (44 out of 65 XOR differences are different in their odd bytes) suitable for the conditions. Then we considered the second set K for all possible (44) $L^\oplus$ differences:

$$K \quad = \quad \{K^+ \in C_{K^\oplus} : K^+ = odd(B_6 \oplus L^\oplus) \boxminus odd(B_6)\}$$

Again the search commenced from the differences of low Hamming weight and the difference $K^\oplus$ of weight 8 could not be found for 17 of the $L^\oplus$ differences. For the remaining 48 $L^\oplus$ values, we searched for the highest cardinality of $|L|$ and $|K|$ at the same time. The following differences are used for $K^\oplus$ and $L^\oplus$.

$$\begin{aligned} L^\oplus &= 82201180A4020104_x \\ K^\oplus &= 9002400040200804_x \end{aligned}$$

Here, it is important that the most significant bits of $L^\oplus$ and $K^\oplus$ are nonzero. We make an extensive use of this fact since $|C_{L^\oplus}| = 2^{11}$ instead of 2^{12} and $|C_{K^\oplus}| = 2^7$ instead of 2^8. According to the XOR differences given above we found $|L| = 2002$ and $|K| = 4$.

The precomputation step of the attack is performed only once. It has a complexity of about 2.2^{32} round computations of `Tiger` (2^{32} round computation for each L and K). This is approximately about $2^{28.5}$ evaluations of the compression function of `Tiger`.

4.3.2 The Attack

Starting from the round 1, as described before, the attack basically consists of imposing differences in state variables until the round 7 by message modification technique

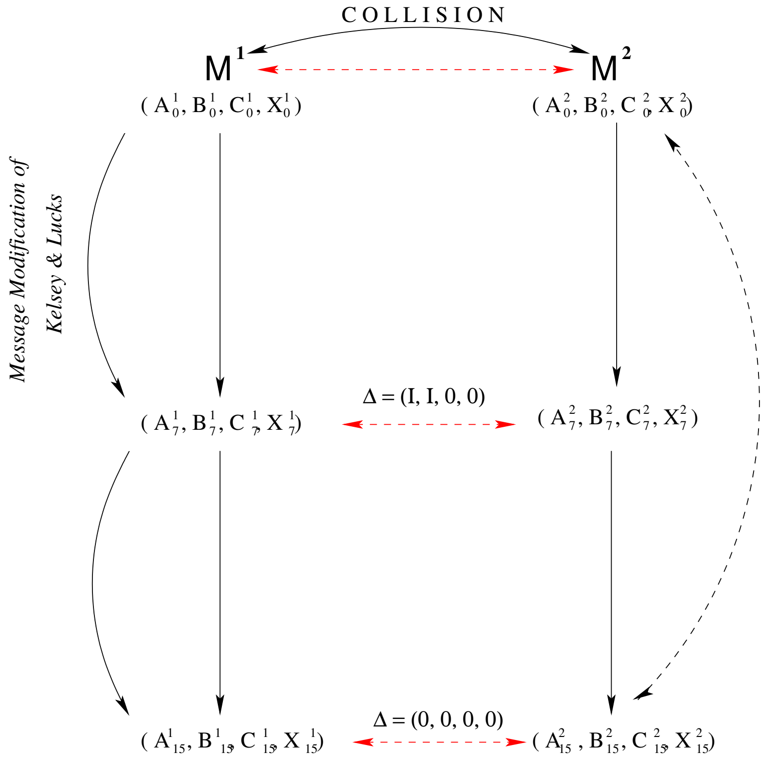


Figure 4.3: Collision Attack on `Tiger-16`

given above, and canceling these differences at the end of round 9. Below, we describe the collision attack on reduced round `Tiger-16` step by step.

1. **Precomputation**: Perform the precomputation step described in the previous subsection. Use $L^{\oplus} = 82201180A4020104_x$ and $K^{\oplus} = 9002400040200804_x$. The differences $L = \{L^+ \in C_{L^{\oplus}} : L^+ = I \boxplus odd(B_7 \oplus I) \boxminus odd(B_7)\}$ and $K = \{K^+ \in C_{K^{\oplus}} : K^+ = odd(B_6 \oplus L^{\oplus}) \boxminus odd(B_6)\}$ are gathered at the end of this step (Also, we have values of B_6 and B_7 satisfying the above criteria). Here $|L| = 2002$ and $|K| = 4$.
   Complexity : The precomputation step of the attack has a complexity of about $2^{28.5}$ evaluations of the compression function of `Tiger`.

2. Choose random values for X_0, X_1 and $X_2[even]$ to compute A_1, B_1, C_1, A_2, B_2, C_2 and C_3.
   Complexity : This step of the attack has a negligible time complexity.

3. Apply a message modification step to construct the XOR-difference $K^{\oplus}$ in round 4. This step needs to be done for all values of K^+ (2^7 times) as message modification works with the modular differences. This step determines the

message words $X_2[odd]$ and $X_3[even]$. Now, we have the values of A_3, B_3, C_3 and C_4 (as well as A_3^*, B_3^*, C_3^* and C_4^*).

Complexity : This step of the attack has a complexity of about $2^{28.5} + (2^7.2^{28.5}).2 \approx 2^{36.5}$ evaluations of the compression function of `Tiger`.

4. Apply a message modification step to construct the XOR-difference $L^{\oplus}$ in round 5. This step needs to be done for all values of $L^+(2^{11}$ times) and determines the message words $X_3[odd]$ and $X_4[even]$. Now, we have the values of A_4, B_4, C_4 and C_5 (as well as A_4^*, B_4^*, C_4^* and C_5^*).

   Complexity : This step of the attack has a complexity of about $2^{36.5} + (2^{11}.2^{28.5}).2 \approx 2^{40.5}$ evaluations of the compression function of `Tiger`.

5. Apply a message modification step to construct the XOR-difference I in round 6. This step of the attack determines the message words $X_4[odd]$ and $X_5[even]$. It is again repeated 2^7 times as we use the modular differences in message modification. Now, we have the values of A_5, B_5, C_5 and C_6 (as well as A_5^*, B_5^*, C_5^* and C_6^*).

   Complexity : This step of the attack has a complexity of about $2^{40.5} + (2^{28.5}.2^7).2 \approx 2^{41}$ evaluations of the compression function of `Tiger`.

6. In order to guarantee that $\Delta^+(B_5)$ can be canceled by $\Delta^+(odd(B_6))$, we need that $\Delta^+(B_5) \in K$. Since we found that $|K| = 4$, this has a probability of $2^{-5} = (4/128)$. In order to guarantee that the difference in B_6 is canceled, we need that $\Delta^+(B_6) \in L$. Since we found that $|L| = 2002$, this has a probability of $2^{0.03}$.

   Complexity : In order to make the desired cancellations in $\Delta^+(B_5)$ the previous computations should be repeated 2^5 times. This step of the attack has a complexity of $2^{41}.2^5 = 2^{46}$ evaluations of the compression function of `Tiger`. Similarly, so as to cancel $\Delta^+(B_6)$ the previous computations should be repeated $2^{0.03}$ times. This step of the attack has a negligible time complexity.

7. Since we know the values of the differences in L and K, we know the values of B_6 and B_7 immediately according to the results of the precomputation step. Therefore, we can determine $X_5[odd]$ and $X_6[odd]$. This adds no additional cost to the attack complexity.

8. Choose $X_6[even]$ and X_7 randomly (obeying to the path) and form the other message words accordingly by using message expansion algorithm. At the end

of this step we are able to compute the message words $M^1 = (X_0^1, ..., X_7^1)$ and $M^2 = (X_0^2, ..., X_7^2)$.

Hence, a collision can be constructed in `Tiger` reduced to 16 rounds with a complexity 2^{46} evaluations of the `Tiger` compression function. Actually, the attack of Kelsey and Lucks has a better complexity, namely 2^{44} evaluations of `Tiger` compression function. They used different techniques for the attack [38]. However, in this slightly modified attack, we present exact values of $L^{\oplus}$ and $K^{\oplus}$.

4.4 Pseudo-Near-Collision Attack on `Tiger`

In this section, we summarize the pseudo-near-collision attack performed by Mendel and Rijmen in [56] to full `Tiger`. This is the first attack mounted on full `Tiger`-hash function. In the attack, difference in the final hash value is the same as in the initial value. Although there is a pseudo-collision after 24 rounds, due to the feed forward the collision after 24 rounds is destroyed. It results in a 1-bit pseudo-near-collision for the `Tiger` hash function.

Mendel and Rijmen used the characteristic given below, for the message expansion of `Tiger`. Instead of the characteristic used by Kelsey and Lucks, this holds with a probability of 2^{-1}.

$$(0, I, 0, 0, 0, I, I', 0) \longrightarrow (0, I, 0, I, 0, 0, 0, 0) \longrightarrow (0, I, 0, 0, 0, 0, 0, 0)$$

Here, as above, I denotes a difference in the most significant bit of the message word and $I' := I >> 23$.

We need to have the following differences in state variables so as to have a pseudo-near-collision in `Tiger`-hash function.

$$\Delta^{\oplus} A_{15} = 0, \Delta^{\oplus} B_{15} = I, \Delta^{\oplus} C_{15} = 0$$

As in the collision attack on `Tiger`-16, the vital part of the attack is to construct the above differences in state variables.

i	ΔA_i	ΔB_i	ΔC_i	ΔX_i
0	I	0	0	0
1	0	0	I	I
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	I
6	*	I	0	I'
7	*	I'	*	0
8	*	*	*	0
9	*	*	*	I
10	*	*	*	0
11	*	*	*	I
12	*	*	$K^{\oplus}$	0
13	0	K^{+}	$L^{\oplus}$	0
14	0	L^{+}	I	0
15	0	I	0	0
16	I	0	0	0
17	0	0	I	I
18	0	0	0	0
19	0	0	0	0
20	0	0	0	0
21	0	0	0	0
22	0	0	0	0
23	0	0	0	0
24	I	0	0	

Table 4.7: The Characteristic for Pseudo-Near-Collision Attack on `Tiger`

4.4.1 The Attack

Starting from round 5, the attack consists of imposing differences in state variables until round 15 by message modification technique given above, and canceling these differences at the end of round 18. Below, we describe the pseudo-near-collision attack on `Tiger` step by step.

1. **Precomputation**: Perform the precomputation step described in the previous sections. Use $L^{\oplus} = 02201080A4020104_x$ and $K^{\oplus} = 0880020019000900_x$. The differences $L = \{L^{+} \in C_{L^{\oplus}} : L^{+} = odd(B_{15} \oplus I) \boxminus odd(B_{15})\}$ and $K = \{K^{+} \in C_{K^{\oplus}} : K^{+} = odd(B_{14} \oplus L^{\oplus}) \boxminus odd(B_{14})\}$ are gathered at the end of this step. Here $|L| = 502$ and $|K| = 2$.

   Complexity : The precomputation step of the attack has a complexity of about $2^{28.5}$ evaluations of the compression function of `Tiger`.

2. Choose random values for B_5 and B_6 and compute $A_6 = (B_5 \boxplus odd(B_6)) \boxtimes mult$. Here, we have $\Delta^+ A_5 = 0, \Delta^+ B_5 = 0, \Delta^+ C_5 = 0$ and $\Delta^+ X_5 = I$, we get $\Delta^+ B_6 = I$ and $\Delta^+(A_6) = A_6 \boxminus A_6$. Note that there is no difference in C_6, since there are no differences in A_5 and $B_6[even]$.

3. Choose a random value for B_7. Since there is a difference in $\Delta^\oplus(X_6) = I'$ and no difference in C_6, we also know the modular difference of $\Delta^+(B_7) = (B_7 \oplus I') \boxminus B_7$. If we have B_7 and B_7^* , by choosing random values for X_7, X_8, X_9 and $X_{10}[even]$, we can calculate B_{10}, C_{10}, C_{11} (and $B_{10}^*, C_{10}^*, C_{11}^*$).
   Complexity : This step of the attack has a negligible time complexity.

4. Apply a message modification step to construct the XOR-difference $K^\oplus$ in round 12. This step needs to be done for all values of $K^\oplus(2^8$ times) and determines the message words $X_{10}[odd]$ and $X_{11}[even]$. Now, we have the values of A_{11}, B_{11}, C_{11} and C_{12} (as well as $A_{11}^*, B_{11}^*, C_{11}^*$ and C_{12}^*).
   Complexity : This step of the attack has a complexity of about $2^{28.5} + (2^8.2^{28.5}).2 \approx 2^{37.5}$ evaluations of the compression function of `Tiger`.

5. Apply a message modification step to construct the XOR-difference $L^\oplus$ in round 13. This step needs to be done for all values of $L^\oplus(2^{10}$ times) and determines the message words $X_{11}[odd]$ and $X_{12}[even]$. Now, we have the values of A_{12}, B_{12}, C_{12} and C_{13} (as well as $A_{12}^*, B_{12}^*, C_{12}^*$ and C_{13}^*).
   Complexity : This step of the attack has a complexity of about $(2^{37.5} + 2^{10}.2^{28.5}).2 \approx 2^{39.5}$ evaluations of the compression function of `Tiger`.

6. Apply a message modification step to construct the XOR-difference I in round 14. This step of the attack determines the message words $X_{12}[odd]$ and $X_{13}[even]$. Now, we have the values of A_{13}, B_{13}, C_{13} and C_{14} (as well as $A_{13}^*, B_{13}^*, C_{14}^*$ and C_{14}^*).
   Complexity : This step of the attack has a complexity of about $(2^{39.5} + 2^{28.5}.2^8).2 \approx 2^{40}$ evaluations of the compression function of `Tiger`.

7. In order to guarantee that $\Delta^+(B_{12})$ can be canceled by $\Delta^+(odd(B_{13}))$, we need that $\Delta^+(B_{12}) \in K$. Since we assume that the Hamming Weight of $K^\oplus$is 8, this has a probability of 2^{-7}. In order to guarantee that the difference in B_{13} is canceled, we need that $\Delta^+(B_{13}) \in L$. Since $L^\oplus$ has a Hamming Weight of 10, this has a probability of 2^{-7}. This determines the message words

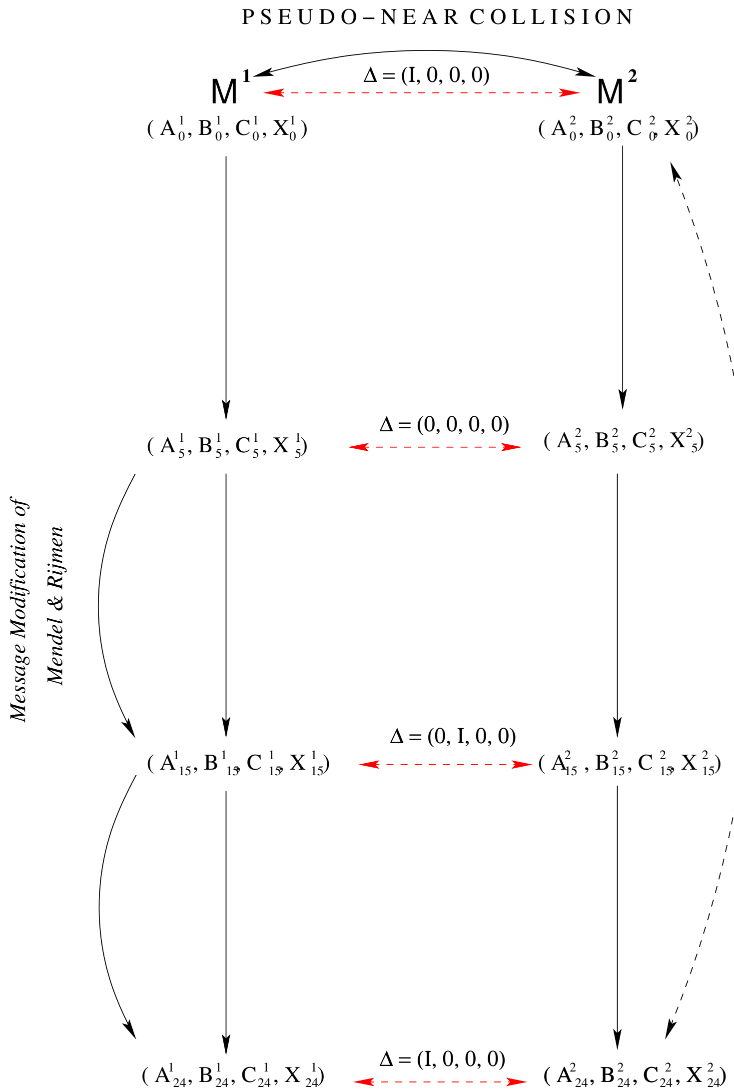


Figure 4.4: Pseudo-Near-Collision Attack on `Tiger`

$X_{12}[odd], X_{13}[even], X_{13}[odd]$ and $X_{14}[odd]$

Complexity : This step of the attack works with the previous step and in order to guarantee the necessary cancellations it needs to be repeated 2^8 times. This step has a complexity of about 2^{48} evaluations of the compression function of `Tiger`.

8. Now we have the message words $X_7, ..., X_{13}$ and $X_{14}[odd]$. To compute the message words $X_0, ..., X_7$ we use the inverse key schedule of `Tiger`. Choose a random value for $X_{14}[even]$ and compute X_{15} as follows:

$$X_{15} = (X_7 \oplus (X_{14} \boxminus X_{13}) \boxminus (X_{14} \oplus 0123456789ABCDEF))$$

Since this step of the attack works with probability 2^{-1}, it should be repeated twice for different values of $X_{14}[even]$. This adds negligible cost to the attack complexity.

9. Compute the `IV` by running the first 8 rounds backwards using $X_0, ..., X_7$.

4.5 Conclusion

In this chapter, the collision search attacks on `Tiger` were presented. Firstly, the collision attack on `Tiger`-16 and then the pseudo-near-collision attack on `Tiger` were mentioned. The attack of Kelsey and Lucks was slightly modified with the exact values of $L^{\oplus}$ and $K^{\oplus}$ and the attack of Mendel and Rijmen was repeated. In the following chapter, we will introduce our related-key boomerang, amplified boomerang and rectangle distinguishers to the encryption mode of `Tiger`.

Chapter 5

Cryptanalysis of the Encryption Mode of `Tiger`

So far, we have investigated the application of the differential cryptanalysis to the several cryptographic primitives, namely block ciphers and hash functions. This chapter is devoted to the application of differential cryptanalysis to the encryption mode of `Tiger`-hash function. We treat `Tiger` as a block cipher and mount a related-key boomerang and related-key rectangle attacks to the reduced (17, 19, 21 rounds out of 24) `Tiger`.

There have been several cryptanalysis papers investigating the randomness properties of the designed hash functions under the encryption modes by such as the paper of Kim *et al* [42]. In that paper, related-key boomerang and related-key rectangle attacks are performed on the encryption modes of `MD4`,`MD5` and `HAVAL` under $2, 4$ related-keys or some weak keys. Moreover, there have been very important attacks [44, 52, 46] on `SHACAL` as well which is based on the hash function `SHA`. Now, we investigate the security notion of reduced round `Tiger` in the encryption mode against the very well known and the efficient block cipher attacks, namely related-key boomerang and the related-key rectangle attacks. Moreover, we present related-key boomerang and rectangle distinguishers of 17, 19 and 21 rounds.

The rest of the chapter is structured as follows. In Section 5.1, we will investigate the security of `Tiger` in the encryption mode and in Section 5.2 we conclude the chapter.

Key Difference	Rounds $0-7$	Rounds $8-15$	Rounds $16-23$
$(I, I, I, I, 0, 0, 0, 0)$ $(0, I, 0, 0, 0, I, I', 0)$	$(I, I, I, I, 0, 0, 0, 0)$ $(0, I, 0, 0, 0, I, I', 0)$	$(I, I, 0, 0, 0, 0, 0, 0)$ $(0, I, 0, I, 0, 0, 0, 0)$	$(., ., ., ., ., ., ., .)$ $(0, I, 0, 0, 0, 0, 0, 0)$*
$(0, I, 0, 0, 0, I, I, I)$ $(0, 0, 0, I, 0, 0, 0, I)$	$(0, I, 0, 0, 0, I, I, I)$ $(0, 0, 0, I, 0, 0, 0, I)$	$(0, 0, 0, 0, 0, I, I, 0)$ $(0, I, 0, 0, 0, 0, 0, I)$	$(0, 0, 0, 0, 0, I, I, I)$ $(0, 0, 0, 0, 0, 0, 0, I)$
$(I, I, 0, 0, 0, I, 0, 0)$ $(0, 0, 0, I, 0, 0, 0, I)$	$(I, I, 0, 0, 0, I, 0, 0)$ $(0, 0, 0, I, 0, 0, 0, I)$	$(0, 0, 0, 0, 0, I, 0, I)$ $(0, I, 0, 0, 0, 0, 0, I)$	$(., ., ., ., ., ., ., .)$ $(0, 0, 0, 0, 0, 0, 0, I)$

Table 5.1: The Propagation of Some Key Differences with probability 1 and 1/2*

5.1 The Related-Key Boomerang and Rectangle Attacks to `Tiger`

In this section, we introduce our related-key boomerang and rectangle distinguishers to reduced round `Tiger`. Two of the distinguishers work with probability one and one of them works with 2^{-1} and they can be easily adapted to the key recovery attacks.

The block cipher mode of `Tiger` is straightforward. The chaining operations of the intermediate values are omitted and `Tiger` is treated as a block cipher encrypting 192-bit plaintext into 192-bit ciphertext using 512-bit secret key. There is no need to invert the *odd* and the *even* functions since their inverses do not affect the decryption mode. In the decryption mode, we just use the inverses of the binary operations that can be defined very easily except for the division *mod* 2^{64}. However, as we divide any number *mod* 2^{64} by an odd constant, this division operation is also well defined. Thus, besides the encryption function, the decryption function is well defined. Moreover, from now on in this section, the message expansion is called the key schedule of `Tiger`.

In `Tiger`, the key scheduling is non-linear. However, some differences propagate linearly. We introduced some of the differentials in previous chapter. Six of them is given in Table 5.1 and used in 17, 19 and 21-round related-key rectangle and boomerang distinguishers. In order to succeed, we need to combine some of these differentials very effectively. Observing the propagation of these differentials, we should make an extensive use of cancellations and probability one differentials as in the collision attacks on `Tiger`. Moreover, low weight differentials and the number of rounds attacked are also very important. In the scope of this simple tricks, the following sections contain our attacks on the encryption mode of `Tiger`.

5.1.1 17-Round Distinguisher

Following the notation introduced in Chapter 3, we treat `Tiger` as a cascade of two sub-ciphers, E_0 and E_1. The rounds between $7-15$ are taken as E_0 and the rounds between $15-23$ are treated as E_1.

The Differential for E_0 (rounds $7-15$)

In `Tiger`, we can find a probability 1 related-key differential for E_0. For E_0, the related-key differential $(I, I, 0) \rightarrow (0, 0, 0)$ works with probability 1 for rounds $7-15$ under the key difference $(I, I, I, I, 0, 0, 0, 0)$ shown in Table 5.1. In round 7, by imposing difference $\alpha = (\Delta A_7, \Delta B_7, \Delta C_7) = (I, I, 0)$, we cancel the subkey difference $\Delta X_8 = I$ with $\Delta C_8 = I$ making $(\Delta A_9, \Delta B_9, \Delta C_9) = (0, 0, I)$. In round 9, as in the previous round, we cancel the subkey difference $\Delta X_9 = I$ with $\Delta C_9 = I$. Finally in round 10, we have $(\Delta A_{10}, \Delta B_{10} \Delta C_{10}) = (0, 0, 0)$. From round 10 until round 15, we use the trivial differential which makes $\beta = (0, 0, 0)$. Notice that, we make an extensive use of the trivial propagation of the I difference through the words B_i and *even* function.

Up to know, everything works with probability 1 and the differential probability p and $\widehat{p}$ for the subcipher E_0 is 1. This is valid for both of the related-key rectangle and the related-key boomerang attacks.

The Differential for E_1 (rounds $15-23$)

For the second part of our distinguisher E_1, the related-key differential $(0, I, 0) \rightarrow (0, 0, 0)$ works with probability 1 for rounds $15-23$ under the key difference $(0, I, 0, 0, 0, I, I', 0)$. Here, according to the notation given in Chapter 3, $\gamma = (0, I, 0)$. Again we will use the trivial propagation of the difference I through the words B_i. The difference γ in round 15 propagates to the round 17 as $(\Delta A_{17}, \Delta B_{17}, \Delta C_{17}) = (0, 0, I)$ with probability 1 and cancels the subkey difference $\Delta X_{17} = I$. From the end of the round 17 till round 23, again we use the trivial differential making $(\Delta A_{23}, \Delta B_{23}, \Delta C_{23}) = (0, 0, 0)$. As in E_0, everything works with probability 1 and the differential probability q and $\widehat{q}$ for the subcipher E_1 is 1. This is valid for both of the related-key rectangle and the related-key boomerang attacks. However, the key scheduling characteristic works with probability 2^{-1} and the attack should be repeated 2 times for different key values.

Round	ΔA	ΔB	ΔC	ΔK	*Probability*
α	I	I	0	K_{12}	
7	I	I	0	0	1
8	I	0	I	I	1
9	0	0	I	I	1
10	0	0	0	0	1
11	0	0	0	0	1
12	0	0	0	0	1
13	0	0	0	0	1
14	0	0	0	0	1
15	0	0	0	0	1
β	0	0	0		
γ	0	I	0	K_{13}	
15	0	I	0	0	1
16	I	0	0	0	1
17	0	0	I	I	1
18	0	0	0	0	1
19	0	0	0	0	1
20	0	0	0	0	1
21	0	0	0	0	1
22	0	0	0	0	1
23	0	0	0	0	1
δ	0	0	0		
REC					$Pr[REC] = 1$
BOO					$Pr[BOO] = 1$

Table 5.2: The characteristic for 17-Round Distinguisher

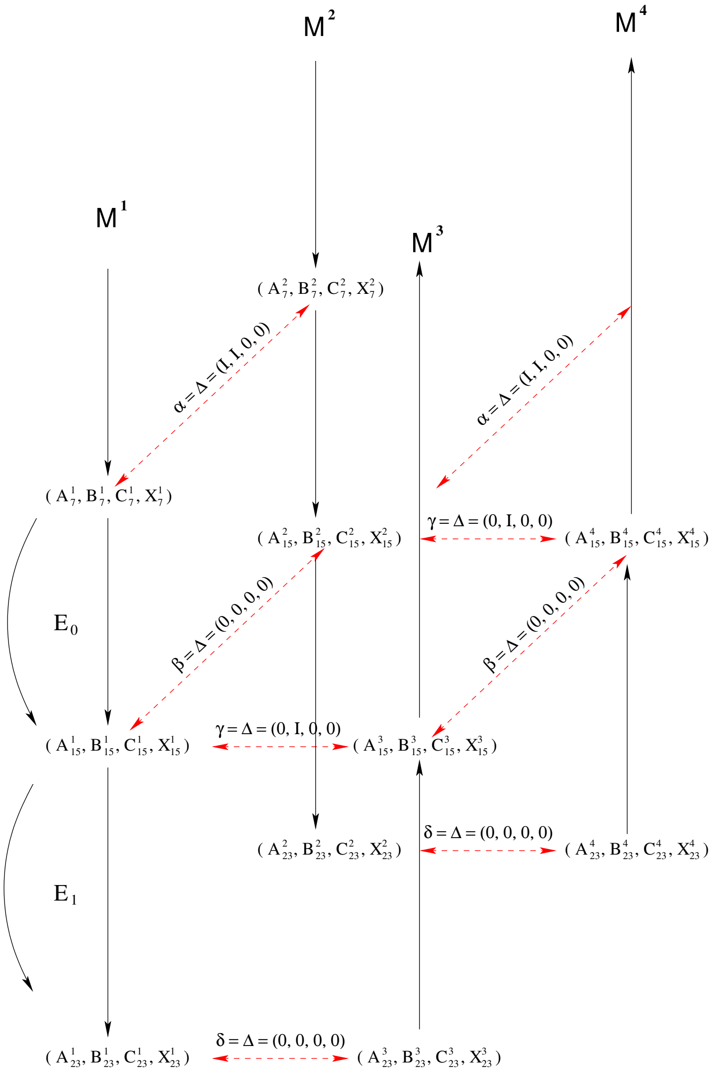


Figure 5.1: 17-Round Related-Key Boomerang Distinguisher for `Tiger`

5.1.2 19-Round Distinguisher

The Differential for E_0 (rounds $5-13$)

As in the 17-round distinguisher, we can find a probability 1 related-key differential for E_0. Here the subcipher E_0 consists of the rounds between 5 and 13. For E_0, the related-key differential $(I, I, I) \rightarrow (0, 0, 0)$ works with probability 1 under the key difference $(0, I, 0, 0, 0, I, I, I)$ shown in Table 5.1. In round 5, by imposing difference $\alpha = (\Delta A_5, \Delta B_5, \Delta C_5) = (I, I, I)$, we cancel the subkey difference $\Delta X_5 = I$ with $\Delta C_5 = I$ making $(\Delta A_6, \Delta B_6, \Delta C_6) = (I, 0, I)$. In round 6, as in the previous round, we cancel the subkey difference $\Delta X_6 = I$ with $\Delta C_6 = I$. Finally in round 7, we have $(\Delta A_7, \Delta B_7, \Delta C_7) = (0, 0, I)$. Again, the subkey difference $\Delta X_7 = I$ and the word C_7 difference $\Delta C_7 = I$ cancel each other. From round 7 until round 13, we use the trivial differential which makes $\beta = (0, 0, 0)$. Notice that, we again make an extensive use of the trivial propagation of the I difference through the words B_i and *even* function as it does not affect the even bytes of the corresponding words.

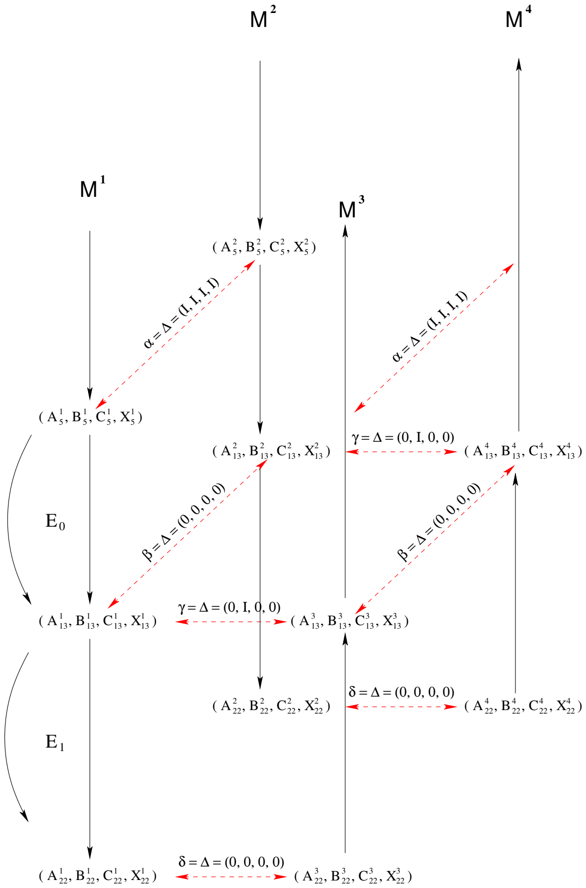


Figure 5.2: 19-Round Related-Key Boomerang Distinguisher for `Tiger`

Up to know, everything works with probability 1 and the differential probability p and $\widehat{p}$ for the subcipher E_0 is 1. This is valid for both of the related-key rectangle and the related-key boomerang attacks.

The Differential for E_1 (rounds $13 - 22$)

For the second part of our distinguisher E_1, the related-key differential $(0, I, 0) \rightarrow (0, 0, 0)$ works with probability 1 for rounds $13 - 22$ under the key difference $(0, 0, 0, I, 0, 0, 0, I)$. Here, according to the notation given in chapter 3, $\gamma = (0, I, 0)$. The difference γ in round 13 propagates to the round 15 as $(\Delta A_{15}, \Delta B_{15}, \Delta C_{15}) =$

Round	ΔA	ΔB	ΔC	ΔK	$Probability$
α	I	I	I	K_{12}	
5	I	I	I	I	1
6	I	0	I	I	1
7	0	0	I	I	1
8	0	0	0	0	1
9	0	0	0	0	1
10	0	0	0	0	1
11	0	0	0	0	1
12	0	0	0	0	1
13	0	0	0	0	1
β	0	0	0		
γ	0	I	0	K_{13}	
13	0	I	0	0	1
14	I	0	0	0	1
15	0	0	I	I	1
16	0	0	0	0	1
17	0	0	0	0	1
18	0	0	0	0	1
19	0	0	0	0	1
20	0	0	0	0	1
21	0	0	0	0	1
22	0	0	0	0	1
δ	0	0	0		
REC					$Pr[REC] = 1$
BOO					$Pr[BOO] = 1$

Table 5.3: The characteristic for 19-Round Distinguisher

$(0, 0, I)$ with probability 1 and cancels the subkey difference $\Delta X_{15} = I$. From the end of the round 15 till round 22, again we use the trivial differential making $(\Delta A_{22}, \Delta B_{22}, \Delta C_{22}) = (0, 0, 0)$. As in E_0, everything works with probability 1 and the differential probability q and $\widehat{q}$ for the subcipher E_1 is 1.

The Round After the Distinguisher

There is also a possibility to add a round after the distinguisher given above. However, this addition is applicable just for the rectangle distinguisher. We have $(\Delta A_{22}, \Delta B_{22}, \Delta C_{22}) = (0, 0, 0)$ and the subkey difference ΔX_{22} in the last round is I. Therefore, the propagation of this difference through the last round leads to the difference $(\Delta A_{23}, \Delta B_{23}, \Delta C_{23}) = (\delta', I, 0)$ where $A_{23} \boxminus A_{23}^* = Odd(B_{23}) \boxminus Odd(B_{23}^*)$. Therefore, the distinguisher works between the rounds $5 - 23$.

5.1.3 21-Round Distinguisher

The Differential for E_0 (rounds $3 - 13$)

Another differential for `Tiger` can be used to extend the distinguisher to 21 rounds. This time the other differential in Table 5.1 is used. In round 3, by imposing difference $\alpha = (\Delta A_3, \Delta B_3, \Delta C_3) = (0, I, 0)$, we cancel the subkey difference $\Delta X_5 = I$ with $\Delta C_5 = I$ making $(\Delta A_6, \Delta B_6, \Delta C_6) = (0, 0, 0)$. From round 6 until round 13, we use the trivial differential which makes $\beta = (0, 0, 0)$.

Again, all differential works with probability one and we make an extensive use of the propagation of I difference through round operations.

The Differential for E_1 (rounds $13 - 22$)

For the second part of our distinguisher E_1, the related-key differential $(0, I, 0) \rightarrow (0, 0, 0)$ works with probability 1 for rounds $13-22$ under the key difference $(0, 0, 0, I, 0, 0, 0, I)$. Here, $\gamma = (0, I, 0)$. Again we will use the trivial propagation of the difference I through the words B_i. The difference γ in round 13 propagates to the round 15 as $(\Delta A_{15}, \Delta B_{15}, \Delta C_{15}) = (0, 0, I)$ with probability 1 and cancels the subkey difference $\Delta X_{15} = I$. From the end of the round 15 till round 22, again we use the trivial differential making $(\Delta A_{22}, \Delta B_{22}, \Delta C_{22}) = (0, 0, 0)$. As in E_0, everything works with probability 1 and the differential probability q and $\widehat{q}$ for the subcipher E_1 is 1. This is

Round	ΔA	ΔB	ΔC	ΔK	$Probability$
α	0	I	0	K_{12}	
3	0	I	0	0	1
4	I	0	0	0	1
5	0	0	I	I	1
6	0	0	0	0	1
7	0	0	0	0	1
8	0	0	0	0	1
9	0	0	0	0	1
10	0	0	0	0	1
11	0	0	0	0	1
12	0	0	0	0	1
13	0	0	0	0	1
β	0	0	0		
γ	0	I	0	K_{13}	
13	0	I	0	0	1
14	I	0	0	0	1
15	0	0	I	I	1
16	0	0	0	0	1
17	0	0	0	0	1
18	0	0	0	0	1
19	0	0	0	0	1
20	0	0	0	0	1
21	0	0	0	0	1
22	0	0	0	0	1
δ	0	0	0		
REC					$Pr[REC] = 1$
BOO					$Pr[BOO] = 1$

Table 5.4: The characteristic for 21-Round Distinguisher

valid for both of the related-key rectangle and the related-key boomerang attacks. As in the previous distinguisher, we can extend the above related-key rectangle distinguisher by adding one round after the distinguisher. Thus, the attack works between the rounds $3-23$.

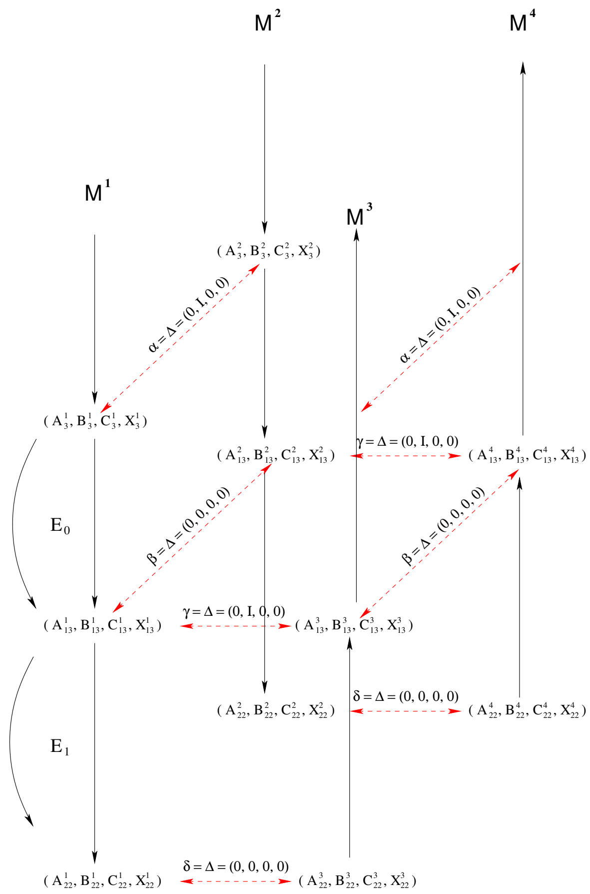


Figure 5.3: 21-Round Related-Key Boomerang Distinguisher for `Tiger`

5.1.4 The Attack

In this subsection, we present our attacks for the last distinguisher as it is similar for the othres. A 20-round related-key boomerang and a 21-round related-key rectangle distinguishing attacks are detailed. For the boomerang distinguisher, we do not use the round after the distinguisher added to the usual related-key boomerang distinguisher that totally covers 21 rounds (17 and 19-round versions are similar). The related key boomerang attack to the reduced round `Tiger` can be summerized as follows:

- Take a randomly chosen plaintext $P_1 = (A_3, B_3, C_3)$ and form $P_2 = (A_3^*, B_3^*, C_3^*)$ as $P_1 \boxminus P_2 = (0, I, 0)$. As the value of P_1 is chosen randomly, P_2 can be chosen accordingly since the differences are known.

- Obtain the corresponding ciphertexts $C_1 = E_{K_1}(P_1)$ and $C_2 = E_{K_2}(P_2)$ through E, where $K_2 = K_1 \boxminus (I, I, 0, 0, 0, I, 0, 0)$.

- Take the second ciphertext pair as $C_3 = C_1$ and $C_4 = C_2$. Since the values of C_1 and C_2 are known, the values of C_3 and C_4 can be arranged accordingly.

- Obtain the corresponding plaintexts $P_3 = E_{K_3}^{-1}(C_3)$ and $P_4 = E_{K_4}^{-1}(C_4)$ through E^{-1}, where $K_3 = K_1 \boxminus (0, 0, 0, I, 0, 0, 0, I)$,
$K_4 = K_3 \boxminus (I, I, 0, 0, 0, I, 0, 0)$.

- Check $P_3 \boxminus P_4 = P_1 \boxminus P_2 = (0, I, 0)$.

- If this is the case, identify the corresponding cipher as `Tiger`.

For the related-key rectangle distinguisher on the other hand, we use the round after the distinguisher added to the related-key rectangle distinguisher that totally covers the rounds $3 - 23$. The 21-round related-key rectangle distinguisher can be given as:

- Prepare 2^{97} randomly chosen plaintexts (P_i, P_i^*) with the prescribed difference α.

- Obtain the corresponding ciphertext $C_1 = E_{K_1}(P_1)$, $C_2 = E_{K_2}(P_2)$, $C_3 = E_{K_3}(P_3)$ and $C_4 = E_{K_4}(P_4)$ where $K_2 = K_1 \boxminus (I, I, 0, 0, 0, I, 0, 0)$ and $K_3 = K_1 \boxminus ((0, 0, 0, I, 0, 0, 0, I)$.

- Check $C_1 \boxminus C_3 = (\delta', I, 0)$ and $C_2 \boxminus C_4 = (\delta', I, 0)$. Note that these differences are not equal to each other. Instead, they are fully determined by the values of C_1, C_2, C_3 and C_4.

- If this is the case identify the corresponding cipher as `Tiger`.

Throughout this chapter, we present several related-key boomerang and rectangle distinguishers to the reduced encryption mode of `Tiger`. However, both attacks given above can be easily generalized to key recovery attacks by guessing subkey values in corresponding rounds. As the encryption mode uses 512-bit secret key, it permits to guess the subkeys of at least 7 rounds. However, it is highly theoretical. The results of the given related-key boomerang distinguishers are given in the appendix. For each related-key-boomerang distinguisher, an example is provided.

Chapter 6

Conclusion

In this work, starting from the differential cryptanalysis we investigated the security of the `Tiger` hash function. Firstly, the theory of the differential cryptanalysis together with some basic applications to `FEAL`-8 and `DES` are studied. In differential cryptanalysis of `FEAL`-8 we made an extensive use of the algorithm that enables us to calculate the differential probability of addition on XOR operation. The application of this algorithm to `FEAL`-8 is obvious and can be generalized to the other block ciphers easily.

Then, the notion was extended to the boomerang, amplified boomerang and rectangle attacks which are known to be some of the most powerful block cipher attacks today. However, for the pure models(the ones using a unique key) of these attacks many block ciphers are known to be secure. For the related-key combined attack versions of these attacks, on the other hand, there exist many important attacks. In this thesis, we applied related-key boomerang and rectangle attacks to the encryption mode of `Tiger` and we have found 17, 19 and 21-round distinguishers. These attacks work with probability 1 and 2^{-1} and can be generalized immediately to the key recovery attacks.

For the original mode of `Tiger`, we recapitulate the collision attacks known today. Firstly, the collision attack of Kelsey & Lucks[38] on `Tiger`-16 is detailed. In this attack, we made some modifications in that we found the exact values of the differences used in their attack which were not exemplified in the original paper (There are also assumptions on this attack in [56]). Moreover, we give the details of the pseudo-near collision attack on `Tiger` [56] which is known to be the first attack on full `Tiger`.

Our real aim was to adopt boomerang-type attacks to find collisions for hash functions. The analogy here is the extension of standard related-key differential characteristic to the related-key boomerang characteristic so as to increase the number of rounds attacked. In this respect, we tried to combine the attacks of Kelsey & Lucks and Mendel & Rijmen. Since the attack of Mendel & Rijmen covers full rounds of `Tiger`, it is not an extension at all. However, ignoring this attack we tried to extend the characteristic of Kelsey & Lucks. The attack can be described as:

1. Apply the message modification of Kelsey and Lucks to the first 8 parts of M_1 and M_2 to impose the differences $\alpha = (\Delta A_7, \Delta B_7, \Delta C_7) = (I, I, 0)$ and gather the message words M_1 and M_2. This will lead to a collision after round 15.

2. Apply the difference $\gamma = (\Delta A_{15}, \Delta B_{15}, \Delta C_{15}) = (0, I, 0)$ to the state variables of M_1 & M_3 and M_2 & M_4 at the end of round 15. Once this is satisfied, by boomerang conditions the message words M_3 and M_4 collide at the end of round 15.

3. Construct the message words M_3 and M_4.

4. From 18-round boomerang distinguisher, it is known there are collisions between the state variables of M_1 & M_3 and M_2 & M_4 at the end of the last round.

However, since the differences between the IVs of M_1 & M_3 and M_2 & M_4 are random, this attack does not succeed. Moreover, the interaction between the 8-round passes do not let us construct the desired message words. Therefore, we concluded that this type of attack is not applicable directly to extend the characteristic. This attack attempt is visualized in Figure 6.1.

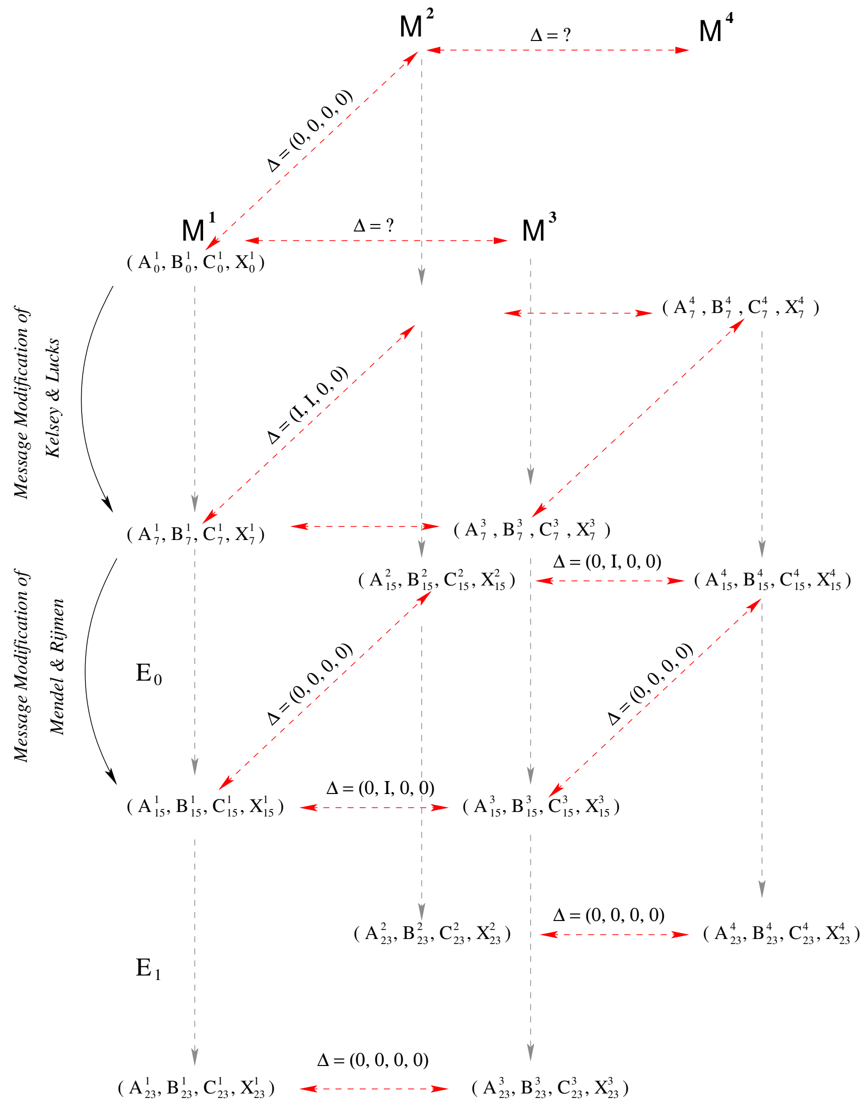


Figure 6.1: An Attempt to Extend the Collision Attack for `Tiger`

References

[1] P.van Oorschot A.Menezes and S.Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.

[2] Ross J. Anderson and Eli Biham. TIGER: A Fast New Hash Function. In Gollmann [31], pages 89–97.

[3] Rana Barua and Tanja Lange, editors. *Progress in Cryptology - INDOCRYPT 2006, 7th International Conference on Cryptology in India, Kolkata, India, December 11-13, 2006, Proceedings*, volume 4329 of *Lecture Notes in Computer Science*. Springer, 2006.

[4] Eli Biham. New Types of Cryptoanalytic Attacks Using related Keys (Extended Abstract). In *EUROCRYPT*, pages 398–409, 1993.

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

[6] Eli Biham and Rafi Chen. Near-Collisions of SHA-0. In *CRYPTO*, pages 290–305, 2004.

[7] Eli Biham, Rafi Chen, Antoine Joux, Patrick Carribault, Christophe Lemuet, and William Jalby. Collisions of SHA-0 and Reduced SHA-1. In Cramer [23], pages 36–57.

[8] Eli Biham, Orr Dunkelman, and Nathan Keller. The Rectangle Attack - Rectangling the Serpent. In Pfitzmann [64], pages 340–357.

[9] Eli Biham, Orr Dunkelman, and Nathan Keller. New Results on Boomerang and Rectangle Attacks. In Daemen and Rijmen [24], pages 1–16.

[10] Eli Biham, Orr Dunkelman, and Nathan Keller. Rectangle Attacks on 49-Round SHACAL-1. In Johansson [34], pages 22–35.

[11] Eli Biham, Orr Dunkelman, and Nathan Keller. New Combined Attacks on Block Ciphers. In Gilbert and Handschuh [30], pages 126–144.

[12] Eli Biham, Orr Dunkelman, and Nathan Keller. Related-Key Boomerang and Rectangle Attacks. In Cramer [23], pages 507–525.

[13] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In Menezes and Vanstone [57], pages 2–21.

[14] Eli Biham and Adi Shamir. Differential Cryptoanalysis of FEAL and N-Hash. In *EUROCRYPT*, pages 1–16, 1991.

[15] Eli Biham and Adi Shamir. Differential Cryptanalysis of the Full 16-Round DES. In Brickell [19], pages 487–496.

[16] Alex Biryukov. *Methods for Cryptanalysis*. PhD thesis, Applied Mathematics Department, Technion, 1999.

[17] Alex 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*. Springer, 2007.

[18] John Black, Martin Cochran, and Trevor Highland. A Study of the MD5 Attacks: Insights and Improvements. In Robshaw [71], pages 262–277.

[19] Ernest F. Brickell, editor. *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*. Springer, 1993.

[20] Christophe De Cannière and Christian Rechberger. Finding SHA-1 Characteristics: General Results and Applications. In *ASIACRYPT*, pages 1–20, 2006.

[21] Anne Canteaut and Kapalee Viswanathan, editors. *Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004, Proceedings*, volume 3348 of *Lecture Notes in Computer Science*. Springer, 2004.

[22] Florent Chabaud and Antoine Joux. Differential Collisions in SHA-0. In Krawczyk [49], pages 56–71.

[23] Ronald Cramer, editor. *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*. Springer, 2005.

[24] Joan Daemen and Vincent 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*. Springer, 2002.

[25] I.B. Damgaard. A Design Principle for Hash Functions. pages 416-427, Springer Verlag, 1990.

[26] Magnus Daum. *Cryptanalysis of Hash Functions of the MD4-Family*. PhD thesis, Ruhr-Universitat Bochum, 2005.

[27] Yvo 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*. Springer, 1994.

[28] Orr Dunkelman. *Techniques for Cryptanalysis of Block Ciphers*. PhD thesis, Computer Science Department, Technion, 2006.

[29] Orr Dunkelman Eli Biham. Differential Cryptanalysis in Stream Ciphers. In *eprint Archive*.

[30] Henri Gilbert and Helena Handschuh, editors. *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*, volume 3557 of *Lecture Notes in Computer Science*. Springer, 2005.

[31] Dieter Gollmann, editor. *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996, Proceedings*, volume 1039 of *Lecture Notes in Computer Science*. Springer, 1996.

[32] Yongfei Han, Tatsuaki Okamoto, and Sihan Qing, editors. *Information and Communication Security, First International Conference, ICICS'97, Beijing, China, November 11-14, 1997, Proceedings*, volume 1334 of *Lecture Notes in Computer Science*. Springer, 1997.

[33] Seokhie Hong, Jongsung Kim, Sangjin Lee, and Bart Preneel. Related-Key Rectangle Attacks on Reduced Versions of SHACAL-1 and AES-192. In Gilbert and Handschuh [30], pages 368–383.

[34] Thomas Johansson, editor. *Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003, Revised Papers*, volume 2887 of *Lecture Notes in Computer Science*. Springer, 2003.

[35] Antoine Joux and Thomas Peyrin. Hash Functions and the (Amplified) Boomerang Attack. In *CRYPTO*, pages 244–263, 2007.

[36] Sokratis K. Katsikas, Javier Lopez, Michael Backes, Stefanos Gritzalis, and Bart Preneel, editors. *Information Security, 9th International Conference, ISC 2006, Samos Island, Greece, August 30 - September 2, 2006, Proceedings*, volume 4176 of *Lecture Notes in Computer Science*. Springer, 2006.

[37] John Kelsey, Tadayoshi Kohno, and Bruce Schneier. Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent. In Schneier [77], pages 75–93.

[38] John Kelsey and Stefan Lucks. Collisions and Near-Collisions for Reduced-Round Tiger. In Robshaw [71], pages 111–125.

[39] John Kelsey, Bruce Schneier, and David Wagner. Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA. In Han et al. [32], pages 233–246.

[40] Jongsung Kim. *Combined Differential, Linear and Related-Key Attacks on Block Ciphers and MAC Algorithms*. PhD thesis, Katholieke Universiteit Leuven, 2006.

[41] Jongsung Kim, Alex Biryukov, Bart Preneel, and Seokhie Hong. On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0 and SHA-1 (Extended Abstract). In Prisco and Yung [67], pages 242–256.

[42] Jongsung Kim, Alex Biryukov, Bart Preneel, and Sangjin Lee. On the Security of Encryption Modes of MD4, MD5 and HAVAL. In Qing et al. [68], pages 147–158.

[43] Jongsung Kim, Seokhie Hong, and Bart Preneel. Related-Key Rectangle Attacks on Reduced AES-192 and AES-256. In Biryukov [17], pages 225–241.

[44] Jongsung Kim, Guil Kim, Seokhie Hong, Sangjin Lee, and Dowon Hong. The Related-Key Rectangle Attack - Application to SHACAL-1. In Wang et al. [81], pages 123–136.

[45] Jongsung Kim, Guil Kim, Sangjin Lee, Jongin Lim, and Jung Hwan Song. Related-Key Attacks on Reduced Rounds of SHACAL-2. In Canteaut and Viswanathan [21], pages 175–190.

[46] Jongsung Kim, Dukjae Moon, Wonil Lee, Seokhie Hong, Sangjin Lee, and Seokwon Jung. Amplified Boomerang Attack against Reduced-Round SHACAL. In Zheng [88], pages 243–253.

[47] Lars R. Knudsen. Truncated and Higher Order Differentials. In Preneel [66], pages 196–211.

[48] Lars 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*. Springer, 1999.

[49] Hugo Krawczyk, editor. *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*. Springer, 1998.

[50] Susan K. Langford and Martin E. Hellman. Differential-Linear Cryptanalysis. In Desmedt [27], pages 17–25.

[51] Helger Lipmaa and Shiho Moriai. Efficient Algorithms for Computing Differential Properties of Addition. In Matsui [54], pages 336–350.

[52] Jiqiang Lu, Jongsung Kim, Nathan Keller, and Orr Dunkelman. Differential and Rectangle Attacks on Reduced-Round SHACAL-1. In Barua and Lange [3], pages 17–31.

[53] Jiqiang Lu, Jongsung Kim, Nathan Keller, and Orr Dunkelman. Related-Key Rectangle Attack on 42-Round SHACAL-2. In Katsikas et al. [36], pages 85–100.

[54] Mitsuru 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*. Springer, 2002.

[55] Florian Mendel, Bart Preneel, Vincent Rijmen, Hirotaka Yoshida, and Dai Watanabe. Update on Tiger. In Barua and Lange [3], pages 63–79.

[56] Florian Mendel and Vincent Rijmen. Cryptanalysis of Tiger Hash Function. In *ASIACRPYT*, 2007.

[57] Alfred Menezes and Scott A. Vanstone, editors. *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, volume 537 of *Lecture Notes in Computer Science*. Springer, 1991.

[58] Shoji Miyaguchi. The FEAL Cipher Family. In Menezes and Vanstone [57], pages 627–638.

[59] Frédéric Muller. Differential Attacks against the Helix Stream Cipher. In Roy and Meier [72], pages 94–108.

[60] Moni Naor, editor. *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, volume 4515 of *Lecture Notes in Computer Science*. Springer, 2007.

[61] National Institute of Standards and Technologies. Secure Hash Standard. In *Federal Information Processing Standards Publication, FIPS-180-1*, April 1995.

[62] National Institute of Standards and Technologies. Secure Hash Standard. In *Federal Information Processing Standards Publication, FIPS-180*, May 1993.

[63] National Institute of Standards and Technologies. Data Encryption Standard. In *Federal Information Processing Standards Publication, FIPS-46-3*, November 1976.

[64] Birgit Pfitzmann, editor. *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *Lecture Notes in Computer Science*. Springer, 2001.

[65] Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Exploiting Coding Theory for Collision Attacks on SHA-1. In Smart [79], pages 78–95.

[66] Bart 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.

[67] Roberto De Prisco and Moti Yung, editors. *Security and Cryptography for Networks, 5th International Conference, SCN 2006, Maiori, Italy, September 6-8, 2006, Proceedings*, volume 4116 of *Lecture Notes in Computer Science*. Springer, 2006.

[68] Sihan Qing, Wenbo Mao, Javier Lopez, and Guilin Wang, editors. *Information and Communications Security, 7th International Conference, ICICS 2005, Beijing, China, December 10-13, 2005, Proceedings*, volume 3783 of *Lecture Notes in Computer Science*. Springer, 2005.

[69] Ron Rivest. The MD4 Message-Digest Algorithm. 1990.

[70] Ron Rivest. The MD5 Message-Digest Algorithm. 1992.

[71] Matthew J. B. Robshaw, editor. *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers*, volume 4047 of *Lecture Notes in Computer Science*. Springer, 2006.

[72] Bimal K. Roy and Willi Meier, editors. *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*. Springer, 2004.

[73] Markku-Juhani Olavi Saarinen. Cryptanalysis of Block Ciphers Based on SHA-1 and MD5. In Johansson [34], pages 36–44.

[74] Yu Sasaki, Lei Wang, Kazuo Ohta, and Noboru Kunihiro. New Message Difference for MD4. In Biryukov [17], pages 329–348.

[75] Martin Schläffer. Cryptanalysis of MD4. Master's thesis, Graz University of Technology, Graz,Austria, 2006.

[76] Martin Schläffer and Elisabeth Oswald. Searching for Differential Paths in MD4. In Robshaw [71], pages 242–261.

[77] Bruce Schneier, editor. *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, volume 1978 of *Lecture Notes in Computer Science*. Springer, 2001.

[78] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. The 128-Bit Block Cipher CLEFIA (Extended Abstract). In Biryukov [17], pages 181–195.

[79] Nigel P. Smart, editor. *Cryptography and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings*, volume 3796 of *Lecture Notes in Computer Science*. Springer, 2005.

[80] David Wagner. The Boomerang Attack. In Knudsen [48], pages 156–170.

[81] Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors. *Information Security and Privacy: 9th Australasian Conference, ACISP 2004, Sydney, Australia, July 13-15, 2004. Proceedings*, volume 3108 of *Lecture Notes in Computer Science*. Springer, 2004.

[82] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the Hash Functions MD4 and RIPEMD. In Cramer [23], pages 1–18.

[83] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In *CRYPTO*, pages 17–36, 2005.

[84] Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Cramer [23], pages 19–35.

[85] Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient Collision Search Attacks on SHA-0. In *CRYPTO*, pages 1–16, 2005.

[86] Hongjun Wu and Bart Preneel. Differential Cryptanalysis of the Stream Ciphers Py, Py6 and Pypy. In Naor [60], pages 276–290.

[87] Hongjun Wu and Bart Preneel. Differential-Linear Attacks Against the Stream Cipher Phelix. In Biryukov [17], pages 87–100.

[88] Yuliang Zheng, editor. *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, volume 2501 of *Lecture Notes in Computer Science*. Springer, 2002.

Appendix A

The Examples of the Attacks

In the attacks usual related-key boomerang distinguisher is used as described below:

- Take a randomly chosen plaintext P_1 and form $P_2 = P_1 \oplus \alpha$.
- Obtain the corresponding ciphertexts $C_1 = E_{K_1}(P_1)$ and $C_2 = E_{K_2}(P_2)$ through E, where $K_2 = K_1 \oplus \Delta K_{12}$.
- Form the second ciphertext pair by $C_3 = C_1 \oplus \delta$ and $C_4 = C_2 \oplus \delta$.
- Obtain the corresponding plaintexts $P_3 = E^{-1}_{K_3}(C_3)$ and $P_4 = E^{-1}_{K_4}(C_4)$ through E^{-1}, where $K_3 = K_1 \oplus \Delta K_{13}$, $K_4 = K_3 \oplus \Delta K_{12}$.
- Check $P_3 \oplus P_4 = \alpha$

The total comlexity of the attacks are negligible and can be performed in a second using an ordinary PC.

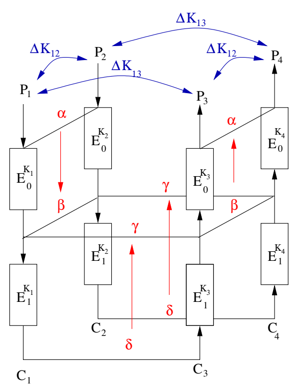


Figure A.1: Related-Key Boomerang Distinguisher Based on Four Related Keys

$P_1 \oplus P_2$	0x8000000000000000, 0x8000000000000000, 0x0000000000000000
P_1	0x9AC1B5074E6EC041, 0x23CB3E897856B783, 0x1E085E27096EC261
P_2	0x1AC1B5074E6EC041, 0xA3CB3E897856B783, 0x1E085E27096EC261
ΔK_{12}	0x8000000000000000, 0x8000000000000000, 0x8000000000000000, 0x8000000000000000, 0x0000000000000000, 0x0000000000000000, 0x0000000000000000, 0x0000000000000000
ΔK_{13}	0x0000000000000000, 0x8000000000000000, 0x0000000000000000, 0x0000000000000000, 0x0000000000000000, 0x8000000000000000, 0x0000010000000000, 0x0000000000000000
K_1	0xCBD055A5A2886272, 0x4B66FB530AB2A11B, 0xC75DAF23D142B034, 0x9157E68BB9D48F85, 0xA4E52A476D8DB9B8, 0xF3B40BFD66AD182D, 0x0AC751DB09010F9A, 0x58F62612CAB5976E
K_2	0x4BD055A5A2886272, 0xCB66FB530AB2A11B, 0x475DAF23D142B034, 0xA55BEBC6E2E1E47E, 0x8B928D6463E1D311, 0xE319E3805A00528C, 0x0AC751DB09010F9A, 0x58F62612CAB5976E
K_3	0xCBD055A5A2886272, 0xCB66FB530AB2A11B, 0xC75DAF23D142B034, 0x9157E68BB9D48F85, 0xA4E52A476D8DB9B8, 0x73B40BFD66AD182D, 0x0AC750DB09010F9A, 0x58F62612CAB5976E
K_4	0x4BD055A5A2886272, 0x4B66FB530AB2A11B, 0x475DAF23D142B034, 0x1157E68BB9D48F85, 0xA4E52A476D8DB9B8, 0x73B40BFD66AD182D, 0x0AC750DB09010F9A, 0x58F62612CAB5976E
C_1	0xD91F598FE1C761BB, 0x17075E71FEE1589C, 0xEE92A40037FB2AAA
C_2	0x74444537E34A238E, 0x0409E3FFBC4D3D54, 0x811C773D2C5576C3
C_3	0xD91F598FE1C761BB, 0x17075E71FEE1589C, 0xEE92A40037FB2AAA
C_4	0x74444537E34A238E, 0x0409E3FFBC4D3D54, 0x811C773D2C5576C3
P_3	0xFCCAE0250E697ABB, 0x10F46BB52B11EB28, 0x0BA8E5FCEFC92625
P_4	0x7CCAE0250E697ABB, 0x90F46BB52B11EB28, 0x0BA8E5FCEFC92625
$P_4 \oplus P_3$	0x8000000000000000, 0x8000000000000000, 0x0000000000000000

Table A.1: An Example to 17-Round Related-Key Boomerang Distinguisher

$P_1 \oplus P_2$	0x8000000000000000, 0x8000000000000000, 0x8000000000000000
P_1	0x5BFEC7BEDF9ACF42, 0xB3B3C30EB3159A93, 0xC5E4033F9A0E5DAB
P_2	0xDBFEC7BEDF9ACF42, 0x33B3C30EB3159A93, 0x45E4033F9A0E5DAB
ΔK_{12}	0x0000000000000000, 0x8000000000000000, 0x0000000000000000, 0x0000000000000000, 0x0000000000000000, 0x8000000000000000, 0x8000000000000000, 0x8000000000000000
ΔK_{13}	0x0000000000000000, 0x0000000000000000, 0x0000000000000000, 0x8000000000000000, 0x0000000000000000, 0x0000000000000000, 0x0000000000000000, 0x8000000000000000
K_1	0x9CAAAE0ED8D1546F, 0xAD8245D21D411DD1, 0x32DF5EBD8FC96B9B, 0xA55BEBC6E2E1E47E, 0x8B928D6463E1D311, 0x6319E3805A00528C, 0xE0B04A6F79B50B1E, 0x11F6E4F3D7C87163
K_2	0x9CAAAE0ED8D1546F, 0x2D8245D21D411DD1, 0x32DF5EBD8FC96B9B, 0x1157E68BB9D48F85, 0xA4E52A476D8DB9B8, 0xF3B40BFD66AD182D, 0x60B04A6F79B50B1E, 0x91F6E4F3D7C87163
K_3	0x9CAAAE0ED8D1546F, 0xAD8245D21D411DD1, 0x32DF5EBD8FC96B9B, 0x255BEBC6E2E1E47E, 0x8B928D6463E1D311, 0x6319E3805A00528C, 0xE0B04A6F79B50B1E, 0x91F6E4F3D7C87163
K_4	0x9CAAAE0ED8D1546F, 0x2D8245D21D411DD1, 0x32DF5EBD8FC96B9B, 0x255BEBC6E2E1E47E, 0x8B928D6463E1D311, 0xE319E3805A00528C, 0x60B04A6F79B50B1E, 0x11F6E4F3D7C87163
C_1	0x23B35087E2A57AF9, 0xE977E99F1B118CB5, 0xEEE039D189EF8E7B
C_2	0x962503719904C7CD, 0x11DEF4450188AFCC, 0xDA8134C11790CBF6
C_3	0x23B35087E2A57AF9, 0xE977E99F1B118CB5, 0xEEE039D189EF8E7B
C_4	0x962503719904C7CD, 0x11DEF4450188AFCC, 0xDA8134C11790CBF6
P_3	0x33ED9A7BB66C5CF2, 0x171F02E113CDCE60, 0x88F74DFE4CFF9D8F
P_4	0xB3ED9A7BB66C5CF2, 0x971F02E113CDCE60, 0x08F74DFE4CFF9D8F
$P_4 \oplus P_3$	0x8000000000000000, 0x8000000000000000, 0x8000000000000000

Table A.2: An Example to 18-Round Related-Key Boomerang Distinguisher

$P_1 \oplus P_2$	0x0000000000000000, 0x8000000000000000, 0x0000000000000000
P_1	0x0B79B8924076136F, 0xA0227877491D319E, 0x221141F4C530C5FE
P_2	0x0B79B8924076136F, 0x20227877491D319E, 0x221141F4C530C5FE
ΔK_{12}	0x8000000000000000, 0x8000000000000000, 0x0000000000000000, 0x0000000000000000, 0x0000000000000000, 0x8000000000000000, 0x0000000000000000, 0x0000000000000000
ΔK_{13}	0x0000000000000000, 0x0000000000000000, 0x0000000000000000, 0x8000000000000000, 0x0000000000000000, 0x0000000000000000, 0x0000000000000000, 0x8000000000000000
K_1	0xA408441DF15EBD84, 0xFA676C8ED9AD179D, 0xE32F5F254DCCD44C, 0xFF85092C483FE5ED, 0x47290A3987C7BD81, 0x2F2A0F08D726A5BA, 0x5505DFA2D1B4EFD1, 0x39F8FD8238E26F7F
K_2	x2408441DF15EBD84, 0x7A676C8ED9AD179D, 0xE32F5F254DCCD44C, 0xFF85092C483FE5ED, 0x47290A3987C7BD81, 0xAF2A0F08D726A5BA, 0x5505DFA2D1B4EFD1, 0x39F8FD8238E26F7F
K_3	0xA408441DF15EBD84, 0xFA676C8ED9AD179D, 0xE32F5F254DCCD44C, 0x7F85092C483FE5ED, 0x47290A3987C7BD81, 0x2F2A0F08D726A5BA, 0x5505DFA2D1B4EFD1, 0xB9F8FD8238E26F7F
K_4	0x2408441DF15EBD84, 0x7A676C8ED9AD179D, 0xE32F5F254DCCD44C, 0x7F85092C483FE5ED, 0x47290A3987C7BD81, 0xAF2A0F08D726A5BA, x5505DFA2D1B4EFD1, 0xB9F8FD8238E26F7F
C_1	0xDCD0C6D011A5B29E, 0x402C4EA1394FCDD8, 0x3F925353D2B2CD95
C_2	0x87EDAF0D1EFB91ED, 0x5BD6EC0499296DF6, 0xCFC8F3081FEF4B63
C_3	0xDCD0C6D011A5B29E, 0x402C4EA1394FCDD8, 0x3F925353D2B2CD95
C_4	0x87EDAF0D1EFB91ED, 0x5BD6EC0499296DF6, 0xCFC8F3081FEF4B63
P_3	0x144F2A412B2E8EFD, 0xA29D4FACFE3B75B2, 0x0D0081CCC3B3ED61
P_4	0x144F2A412B2E8EFD, 0x229D4FACFE3B75B2, 0x0D0081CCC3B3ED61
$P_4 \oplus P_3$	0x0000000000000000, 0x8000000000000000, 0x0000000000000000

Table A.3: An Example to 20-Round Related-Key Boomerang Distinguisher