IMPOSSIBLE DIFFERENTIAL CRYPTANALYSIS OF REDUCED ROUND HIGHT

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

BY

CİHANGİR TEZCAN

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

JULY 2009

Approval of the thesis:

IMPOSSIBLE DIFFERENTIAL CRYPTANALYSIS OF REDUCED ROUND HIGHT

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

Prof. Dr. Ersan Akyıldız _____
Director, Graduate School of **Applied Mathematics**

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

Assoc. Prof. Dr. Ali Doğanaksoy _____
Supervisor, **Mathematics**

Examining Committee Members:

Prof. Dr. Ersan Akyıldız _____
METU, Institute of Applied Mathematics

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

Dr. Muhiddin Uğuz _____
METU, Department of Mathematics

Dr. Meltem Sönmez Turan _____

Dr. Nurdan Saran _____
Çankaya University, Department of Computer Engineering

Date: _____

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

Name, Last Name: CİHANGİR TEZCAN

Signature :

ABSTRACT

IMPOSSIBLE DIFFERENTIAL CRYPTANALYSIS OF REDUCED ROUND HIGHT

Tezcan, Cihangir
M.Sc., Department of Cryptography
Supervisor : Assoc. Prof. Dr. Ali Doğanaksoy

July 2009, 49 pages

Design and analysis of lightweight block ciphers have become more popular due to the fact that the future use of block ciphers in ubiquitous devices is generally assumed to be extensive. In this respect, several lightweight block ciphers are designed, of which HIGHT is proposed by Hong et al. at CHES 2006 as a constrained hardware oriented block cipher. HIGHT is shown to be highly convenient for extremely constrained devices such as RFID tags and sensor networks and it became a standard encryption algorithm in South Korea.

Impossible differential cryptanalysis is a technique discovered by Biham et al. and is applied to many block ciphers including Skipjack, IDEA, Khufu, Khafre, HIGHT, AES, Serpent, CRYPTON, Twofish, TEA, XTEA and ARIA. The security of HIGHT against impossible differential attacks is investigated both by Hong et al. and Lu: An 18-round impossible differential attack is given in the proposal of HIGHT and Lu improved this result by giving a 25-round impossible differential attack. Moreover, Lu found a 28-round related-key impossible differential attack which is the best known attack on HIGHT. In related-key attacks, the attacker is assumed to know the relation between the keys but not the keys themselves.

In this study, we further analyzed the resistance of HIGHT against impossible differential at-

tacks by mounting a new 26-round impossible differential attack and a new 31-round related-key impossible differential attack. Although our results are theoretical in nature, they show new results in HIGHT and reduce its security margin further.

Keywords: Impossible Differential Cryptanalysis, Related-key Attacks, HIGHT, Lightweight Block Ciphers

ÖZ

DÖNGÜ SAYISI AZALTILMIŞ HIGHT BLOK ŞİFRESİNİN İMKANSIZ DİFERANSİYEL KRİPTANALİZİ

Tezcan, Cihangir
Yüksek Lisans, Kriptografi Bölümü
Tez Yöneticisi : Doç. Dr. Ali Doğanaksoy

Haziran 2009, 49 sayfa

Yakın gelecekte blok şifrelerin hafif platformlarda sıkça kullanılacağı hemen hemen herkes tarafından öngörülmektedir. Bu sebeple özellikle son dönemlerde bu tip platformlara uygun blok şifrelerin tasarım ve analizi çokça rağbet görmektedir. Bu şifrelerden birisi olan HIGHT, Hong vd. tarafından CHES 2006'da kısıtlı donanımlara yönelik bir blok şifre olarak sunulmuştur. HIGHT'ın, radyo frekansı ile tanımlama (RFID) etiketleri ve algı ağları gibi son derece kısıtlı cihazlar için çok elverişli olduğu gösterilmiş ve HIGHT Güney Kore'de bir standart şifreleme algoritması olmuştur.

İmkansız difereansiyel kriptanaliz tekniği Biham vd. tarafından bulunmuş ve Skipjack, IDEA, Khufu, Khafre, HIGHT, AES, Serpent, CRYPTON, Twofish, TEA, XTEA ve ARIA gibi bir çok blok şifreye uygulanmıştır. HIGHT'ın imkansız diferansiyel saldırılarına karşı güvenliği Hong vd. ve Lu tarafından incelenmiştir: Hong vd. HIGHT'ı sundukları çalışmalarında 18 döngülük bir imkansız diferansiyel atak tanımlamışlardır ve Lu 25 döngülük yeni bir imkansız diferansiyel atak vererek bu sonucu geliştirmiştir. Ayrıca Lu'nun sunduğu 28 döngülük ilişkili iki anahtar kullanan imkansız diferansiyel atağı, HIGHT'a yapılan şu ana kadarki en iyi

ataktır. İlişkili anahtar saldırılarında anahtarlar arasındaki ilişki bilinmektedir.

Bu çalışmada HIGHT'ın imkansız differensiyal saldırılarına karşı olan dayanıklılığını yeni bir 26 döngülük imkansız diferansiyel atak ve yeni bir 31 döngülük ilişkili anahtar imkansız diferansiyel atak vererek inceledik. Bu saldırılar teorik ataklar olmalarına rağmen, HIGHT için yeni sonuçlar göstermekte ve HIGHT'ın güvenlik sınırını azaltmaktadır.

Anahtar Kelimeler: İmkansız Diferansiyel Kriptanaliz, İlişkili Anahtar Saldırıları, HIGHT, Hafif Blok Şifreler

To my father Hayri, my mother Sırma and my brother Cem

ACKNOWLEDGMENTS

First of all, I would like to express my gratitude to my instructors at Department of Mathematics: Assoc. Prof. Ali DOĞANAKSOY, Assist. Prof. Feza ARSLAN, Assist. Prof. Özgür KİŞİSEL, Dr. Muhammed DABBAGH, Assist. Prof. Ayşe BERKMAN, Prof. Dr. Tanıl ERGENÇ, Prof. Dr. Cem TEZER, Prof. Dr. Hurşit ÖNSİPER, and Prof. Dr. Zafer NURLU, for making me realize the unexpected beauty of mathematics.

It is a great pleasure to thank my colleagues and co-authors Kerem VARICI and Onur ÖZEN for their support and motivation. This study would not be complete without their help.

I would like to thank Çağdaş ÇALIK and Meltem Sönmez TURAN for proofreading the thesis.

I am grateful to my family for always trusting and supporting me. I want to thank my friends and everyone at Institute of Applied Mathematics.

Finally, I would like to thank The Scientific and Technological Research Council of Turkey (TÜBİTAK) for supporting me with M.Sc. scholarship.

TABLE OF CONTENTS

ABSTRACT iv

ÖZ vi

DEDICATION viii

ACKNOWLEDGMENTS ix

TABLE OF CONTENTS x

LIST OF TABLES xii

LIST OF FIGURES xiii

CHAPTERS

1 INTRODUCTION 1

1.1 Block Ciphers 2

1.2 Cryptanalysis of Block Ciphers 3

1.3 Complexity 4

1.4 Differential Cryptanalysis 5

1.5 Related-Key Attacks 5

1.6 Impossible Differential Cryptanalysis 6

1.7 Related-Key Impossible Differential Cryptanalysis 7

1.8 Our Contribution and the Structure of the Thesis 8

2 OVERVIEW OF HIGHT 10

2.1 HIGHT 10

2.2 Notation 11

2.3 Specifications 12

2.4 Previous Impossible Differential Attacks on HIGHT 14

2.4.1 Attack on 18-round HIGHT 15

2.4.2 Attack on 25-round HIGHT 15
2.5 Previous Related-key Impossible Differential Attacks on HIGHT . . 17
2.5.1 Attack on 28-round HIGHT 17
3 IMPOSSIBLE DIFFERENTIAL CRYPTANALYSIS 19
3.1 17-round Impossible Differential Characteristic 19
3.2 26-round Path 21
3.3 Data Collection and Memory 22
3.4 Impossible Differential Attack on HIGHT-26 24
3.5 Complexity of the Attack 29
3.5.1 Data Complexity 29
3.5.2 Memory Complexity 29
3.5.3 Time Complexity 29
3.6 Summary 30
4 RELATED-KEY IMPOSSIBLE DIFFERENTIAL CRYPTANALYSIS 32
4.1 22-round Related-key Impossible Differential Characteristic 32
4.2 31-round Path 33
4.3 Data Collection 34
4.4 Related-Key Impossible Differential Attack on HIGHT-31 36
4.5 Complexity 39
4.5.1 Data Complexity 40
4.5.2 Memory Complexity 40
4.5.3 Time Complexity 41
4.6 Summary 42
5 CONCLUSION 44
REFERENCES 46
APPENDICES

LIST OF TABLES

TABLES

Table 2.1 Comparison of the hardware implementations of HIGHT and AES 11

Table 2.2 Notation 11

Table 2.3 Relations Between the Original Key and Whitening Keys and Subkeys . . . 13

Table 2.4 13-Round Impossible Differential Characteristic 16

Table 2.5 16-Round Impossible Differential Characteristic 16

Table 2.6 19-Round Related-key Impossible Differential Characteristic, $\Delta MK_{10} = e_7$ 18

Table 3.1 17-Round Impossible Differential Characteristic 20

Table 3.2 26-Round Impossible Differential Path 23

Table 3.3 26-Round impossible differential attack 28

Table 3.4 Time Complexities of Each Step of the Attack 31

Table 3.5 Summary of the impossible differential attacks on HIGHT 31

Table 4.1 22-Round Related-key Impossible Differential Characteristic, $\Delta MK_9 = e_7$. 33

Table 4.2 31-Round Related-Key Impossible Differential 35

Table 4.3 31-Round related key impossible differential attack, $\Delta MK_{15} = e_7$ 40

Table 4.4 Time Complexities of Each Step of the Attack 42

Table 4.5 Summary of the related-key impossible differential attacks on HIGHT . . . 43

Table 5.1 Summary of the known attacks on HIGHT 45

LIST OF FIGURES

FIGURES

Figure 2.1 *i*th Round of HIGHT 12

CHAPTER 1

INTRODUCTION

Cryptology, the science of communication secrecy consists of two main components, *cryptography* and *cryptanalysis*. Cryptography is the science of designing secure ciphers and cryptanalysis is the science of analysing the security of ciphers by trying to find weaknesses in the design.

A cipher makes a message unreadable to anyone except those having the key by using an algorithm. More formally, let P denote the *message space*, which contains strings of symbols of a predetermined alphabet and C denote the *ciphertext space* which also contains strings of symbols of a predetermined alphabet.

An element p of P is called a *plaintext* and an element c of C is called a *ciphertext*. Let K denote the *key space* that contains strings of predetermined size. An element k of K is called a *key*. A one-to-one function E_e from P to C, which is uniquely determined by e is called an *encryption function*. One-to-one property is necessary since we want to reverse the process. A one-to-one function D_d from $E_e(P) \subset C$ to P, which is uniquely determined by d is called a *decryption function*.

A *cipher* or an *encryption scheme* contains an encryption function E_e and a decryption function D_d where $e, d \in K$ and d is uniquely determined for any e.

If e and d are equivalent or one of them can be easily obtained from the other in a cipher, that scheme is called a *symmetric-key scheme* or *symmetric-key encryption*. Two main symmetric-key encryption schemes are *block ciphers* and *stream ciphers*.

1.1 Block Ciphers

In a block cipher, the message p is divided into fixed length strings which are called *blocks* and one block is encrypted at a time. Generally, the encryption is done by iterating the *round function* of the cipher for r many times where r is a predetermined integer.

Theory of block ciphers is well investigated and a lot of block ciphers are proposed. Although most of these block ciphers have different designs, they can be roughly categorized as *Feistel Networks* and *Substitution-Permutation Networks (SPNs)*.

In Feistel networks, a round consists of dividing the input into two halves, applying the round function to one half using a subkey, exclusive-oring (XOR) the output of the round function with the other half and swapping the two halves. There is no need to do the swapping operation in the last round since it would not have any effect on the security of the cipher.

Encryption and decryption is identical in Feistel networks except for the order of the subkeys. A Feistel cipher is called *unbalanced* if the divided parts are not of equal size and this kind of constructions are investigated in [32].

SPN uses substitution boxes (S-boxes) and permutation boxes (P-boxes) where an nxm S-box substitutes n bits with m bits and a P-box permutes the bits. Generally in a SPN a round consists of XORing the input with a subkey, applying S-boxes and then P-boxes. The output of the last round is also XORed with a subkey.

The Feistel network is named after Horst Feistel who did important research in this area and proposed Lucifer [31] cipher with Don Coppersmith which is a Feistel network. The Data Encryption Standard (DES) [26], which is a revised version of Lucifer algorithm, is designed by an IBM team in 1974 and it is adopted as national standard in 1977 by National Bureau of Standards (which is known as National Institute of Standards and Technology (NIST) today). DES is an example of a Feistel cipher.

Since the advances in technology result in faster central processing units, 56-bit key of DES was becoming vulnerable to brute force attacks in which the attacker tries every possible key. For this reason, on January 2, 1997, NIST announced a request for candidate algorithms for the Advanced Encryption Standard (AES) [18] which would support 128, 192 and 256-bit keys. 15 algorithms were submitted to the competition and on October 2, 2000, NIST

announced [19] that the winner of the AES competition is Rijndael [1], which is designed by Daemen and Rijmen. AES is an example of an SPN.

1.2 Cryptanalysis of Block Ciphers

One might choose to keep the encryption algorithm secret to increase the security. However in history, it is observed that secret algorithms obtained by reverse engineering, betrayal and espionage. Hence it is a good idea to assume that the security of the encryption algorithm should rely on the secrecy of the key, which is also know as the *Kerckhoffs' Principle*.

The most trivial way to attack a block cipher is to try every key in the key space. This is known as *exhaustive search* or *brute force attack*. This can be done by obtaining a few plaintexts and corresponding ciphertexts and encrypting these plaintexts by every possible key (this makes the attack a known-plaintext attack). If a key encrypts the plaintext to the previously known ciphertext, then that key becomes a candidate but sometimes it may not be the correct secret key. Such a case is called a *false alarm*. This is why we need to test that candidate key on more than one plaintext. A similar attack can be done by only decrypting the ciphertexts and checking whether the obtained plaintexts are something meaningful in the language that the plaintexts are suspected to be written. In that sense the attack becomes a ciphertext-only attack.

Note that exhaustive search is a generic method and it can be applied to any block cipher. For this reason the key space is kept large in the design of block ciphers to avoid brute force attacks. The length of the key depends on the computational power of computers which depends on the current technology.

If the block size of a block cipher is b and if we know 2^b plaintexts and corresponding ciphertexts, this means that although we do not know the key, given a plaintext we can find the corresponding ciphertext or vice versa. This generic attack is known as *dictionary attack* or *table attack*.

A cipher is considered *broken* if an attack is given which finds the secret key faster than exhaustive search and uses less data than dictionary attack. If the attack is still infeasible, it is called a *theoretical attack* and otherwise, it is called a *practical attack*.

Most of the attacks in the literature requires additional knowledge about the system which determines the type of the attack:

- **Ciphertext-only attack (CO):** In this kind of attacks, the attacker has the knowledge of some ciphertexts which are encrypted by the unknown secret key.
- **Known-plaintext attack (KP):** In known-plaintext attacks, the attacker has the knowledge of some plaintexts and corresponding ciphertexts.
- **Chosen-plaintext (ciphertext) attack (CP):** In this scenario, the attacker has the knowledge of some plaintexts having some particular structure of her choice and corresponding ciphertexts. Similarly for chosen-ciphertext attack, the attacker has the knowledge of some ciphertexts having some particular structure of her choice and corresponding plaintexts.
- **Adaptive chosen-plaintext (ciphertext) attack (ACP):** This type of attack is similar to chosen plaintext attack. However this time, the chosen plaintexts depends on the results of the previous encryptions of plaintexts. Similarly for adaptive chosen-ciphertext attack, the chosen ciphertexts depends on the results of the previous decryptions of ciphertexts.

1.3 Complexity

Attacks are compared with the amount of resources they require. These resources are defined as data complexity, time complexity and memory complexity:

- **Data Complexity:** The amount of plaintexts or ciphertexts that is required to perform the attack.
- **Memory Complexity:** The amount of storage required to perform the attack.
- **Time Complexity:** The amount of time required to perform the attack. Most of the time, it is measured by the number of encryptions or memory accesses.

In the case of exhaustive search, if the secret key is n bits, then the time complexity is 2^n encryptions. The attack requires a few plaintexts or ciphertexts and the storage is required

only for the values that are used in the encryption (or decryption) process. Hence the data and memory complexities are negligible.

In the case of dictionary attack, if we put every ciphertext and plaintext in a table, such an attack has 2^b data complexity, 2^b memory complexity and negligible time complexity.

1.4 Differential Cryptanalysis

Differential cryptanalysis [25] was discovered by Biham and Shamir in late 1980s and it is used to attack various block ciphers, stream ciphers and hash functions. This technique breaks DES theoretically.

Differential cryptanalysis is a statistical chosen-plaintext attack and it considers differential relations between inputs and outputs for r consecutive rounds, for some integer r.

When two different inputs are encrypted with the secret key, the probability of the difference of the corresponding outputs to be β, for some β, is 2^{-b} where b is the block size. If an α difference in input blocks results in β difference in the output blocks after r rounds of encryption with a probability higher than 2^{-b}, we call this relation an r-round differential *characteristic*. A differential characteristic with high probability is used to distinguish the correct subkeys from the wrong ones.

Today, differential cryptanalysis plays an important role in the design of blocks ciphers and designers make their algorithms resistant to this attack by giving an upper bound to the probability of r-round differentials [27].

In 1994, Knudsen discovered truncated differential cryptanalysis [35] which is an extension of differential cryptanalysis in which the differences are not fully specified.

1.5 Related-Key Attacks

In related key attacks, the adversary knows the relation between the keys that are used but does not know the keys. If the key scheduling part of a block cipher is not strong, this additional knowledge of the relation between the keys reduces the strength of the whole block cipher. It is important to use these kind of attacks to find weaknesses in the key scheduling part of the

algorithm.

The idea of related-key attacks are found independently by Biham [11] and Knudsen [28]. In [11, 28], it is shown that LOKI89 [29], LOKI91 [30] and Lucifer ciphers have weaknesses for related key attacks.

1.6 Impossible Differential Cryptanalysis

The cryptanalytic technique of impossible differential attack is discovered by Biham, Biryukov and Shamir and it is first presented at Rump Session of CRYPTO 1998 by Shamir [16]. Later on in [14], they presented this technique by giving an attack that breaks Skipjack [17] reduced from 32 to 31 rounds. They also used this technique to break reduced round version of IDEA [21] and Khufu [23] in [22]. Independently in 1998, in his proposal [20] for AES, Knudsen gave an attack to 6-round DEAL [20] which is similar to impossible differential cryptanalysis.

Impossible differential cryptanalysis uses an impossible differential which is a truncated differential characteristic that holds with probability 0. One way of obtaining such a differential is the *miss in the middle* technique, that is the combination of two truncated differentials both of which hold with probability 1 and do not meet in the middle. That is, if a difference α becomes β after r_1 rounds of encryption and a difference δ becomes γ after r_2 rounds of decryptions and if $\beta \neq \gamma$, we conclude that the difference α cannot become δ after $r_1 + r_2$ rounds of encryption. i.e. $\alpha \rightarrow \beta \neq \gamma \leftarrow \delta$.

An impossible differential obtained with a miss in the middle technique works as a *sieve* in the procedure. If under a subkey that impossible differential characteristic holds, it means that the corresponding subkey is not the correct subkey and we eliminate it.

As in the case of differential cryptanalysis, impossible differential attacks are chosen plaintext attacks.

An impossible differential characteristic on r rounds of a block cipher can be used to distinguish a random permutation f from r-round version of that cipher. Assume the α difference cannot produce β after r rounds of encryption. If an input pair of f has the difference α and the corresponding output difference is β, then it is obvious that f is not the r-round version of the cipher. If difference β is not observed, the number of pairs should be increased enough to

be sure that f is not a random permutation. This number of pairs depends on the block size of the cipher and the structure of the characteristic.

For example, assume that the block size is n and the β difference has fixed k bits (hence $n-k$ bits of β can be anything). For a random pair, the probability of not observing the β difference is $1-2^{-k}$. Therefore if we use 2^p pairs and do not observe the β difference, the probability of incorrectly identifying a random permutation as the r-round version of the cipher is $(1-2^{-k})^{2^p}$. Hence, we must choose the value of p larger than k to make this probability close to 0. For small values of k, this probability can be calculated easily with a computer. For large values, the following approximation can be used:

$$\left(1-\frac{1}{2^k}\right)^{c\cdot 2^k} = \left(\left(1-\frac{1}{2^k}\right)^{2^k}\right)^c \approx \left(\frac{1}{e}\right)^c = \frac{1}{e^c}. \tag{1.1}$$

This approximation can be obtained by substituting -1 for x in the formal limit definition of exponential function which is

$$e^x = \lim_{n\to\infty}\left(1+\frac{x}{n}\right)^n. \tag{1.2}$$

When constructing plaintext pairs, the idea of *structures* is generally used in differential attacks. Assume that the α difference has x many 0 bits and y many undetermined bits and the block size n is $x+y$. Fixing the bits in the places where α has no difference is called a *structure*. Now we can construct 2^y different blocks which is in this structure and any two blocks of this structure has difference α. Hence the maximum number of pairs we can obtain from this structure is

$$\binom{2^y}{2} = \frac{2^y\cdot(2^y-1)}{2} = 2^{2y-1}-2^{y-1} \approx 2^{2y-1}. \tag{1.3}$$

Since we have 2^x different structures, approximately $2^{2y-1}\cdot 2^x = 2^{2y+x-1}$ different pairs can be constructed at most. Note that if we replace one or more of the zeros with ones in the difference α, the maximum number of pairs that can be obtained becomes exactly 2^{2y+x-1}. This is the case in the attacks that we are going to define in Chapter 3 and Chapter 4.

1.7 Related-Key Impossible Differential Cryptanalysis

In [37], Kelsey et al. introduced related-key differential attacks in which the attacker either knows or chooses the difference in the keys. However, this is a very strong requirement and

this attack scenario does not concern most of the practical applications.

The impossible differential cryptanalysis uses an impossible differential characteristic of r rounds. The number of rounds that is attacked and the complexity of the attack directly depend on r and the structure of the characteristic. The attacker desires to find longer characteristics with lots of indetermined bits. The design of the block cipher determines the upper bound for this r for the miss in the middle technique. However, a knowledge of difference in the keys may make it possible to obtain related-key impossible differential characteristics of more rounds than this upper bound (i.e. there is another upper bound for the case of related keys).

If the key scheduling part of an algorithm contains weaknesses, it may become possible to attack more rounds with related key impossible differential attacks than impossible differential attacks.

1.8 Our Contribution and the Structure of the Thesis

HIGHT [2, 3] is a variant of generalized Feistel network block cipher which has become a standard encryption algorithm in South Korea [3]. Security of HIGHT against known attacks is investigated in [2] and an impossible differential attack on 18-round HIGHT is found. Lu also investigated HIGHT's security in [13, 12] and presented an impossible differential attack and a related-key impossible differential attack that can be applied to 25 and 28-round HIGHT, respectively.

In this thesis, we further investigate HIGHT's security and present an impossible differential attack on 26-round HIGHT which has a better time complexity than Lu's 25-round attack and we also present a related-key impossible differential attack on 31-round HIGHT which is slightly better than the exhaustive search.

This thesis is organized as follows: In Chapter 2, we first describe the block cipher HIGHT and then we briefly mention the previous impossible differential and related-key impossible differential attacks on HIGHT. In Chapter 3, we investigate HIGHT's security against impossible differential attacks. We first give a 17-round impossible differential characteristic, which is the highest round impossible differential characteristic we could find, and then present our impossible differential attack on 26-round HIGHT which uses a 16-round impossible differ-

ential characteristic. In Chapter 4, we present our related-key impossible differential attack on 31-round HIGHT which uses a 22-round related-key impossible differential characteristic. Finally, we conclude our thesis with Chapter 5.

CHAPTER 2

OVERVIEW OF HIGHT

2.1 HIGHT

Lightweight cryptography has become very vital with the emerging needs in sensitive applications like radio-frequency identification (RFID) systems and sensor networks. For these types of special purposes, there is a strong demand in designing secure lightweight cryptographic modules. After the selection of AES, the research on efficient implementation of AES, especially for such constrained environments, brought special attention in research community. AES is not suitable for extremely constrained devices and the research on designing and analyzing new lightweight block ciphers that are more efficient than AES on these platforms poses huge challenges as block ciphers are assumed to be widely used in ubiquitous devices.

For this purpose, several block ciphers are designed as potential candidates such as HIGHT [2, 3], PRESENT [4], mCrypton [5], SEA [6], CGEN [7], DESL [8] and DESXL [8] (TEA [9] and XTEA [10] can also be given as lightweight block ciphers which were designed before AES).

HIGHT, as one of these candidates enjoying the use of a low-resource hardware implementation, is a 32 round block cipher proposed at Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2006 by Deukjo Hong et al. [2].

The comparison of the hardware implementation of HIGHT with AES, which is done by the designers of HIGHT, is given in Table 2.1 which shows that HIGHT can be implemented with 3048 gates and is much faster than the compared implementation of AES.

In a recent study [33], Lim et al. proposed an implementation of HIGHT for an RFID tag

Table 2.1: Comparison of the hardware implementations of HIGHT and AES

Algorithm	Technology (μm)	Area (GEs)	throughput (Mbps)	Max frequency (MHz)
AES [15]	0.35	3400	9.9	80
HIGHT [2]	0.25	3048	150.6	80

which eliminates the redundant logics and gives more efficient key schedule design. This design has 2608 gates and it is 13% smaller than the original HIGHT design excluding decryption block. Moreover this new design shows outstanding results in power and performance.

In [34], Rinne et al. made performance analysis of lightweight block ciphers DESL, HIGHT, SEA, TEA and XTEA on 8-bit microcontrollers and compared them with AES. They showed that HIGHT's performance is better than the others but requires more memory. They also showed that AES outperforms the others in terms of throughput.

At 27 December 2006, "64-bit Block Cipher HIGHT" was made a standard encryption algorithm in South Korea [3] by Telecommunications Technology Associations (TTA), South Korea, with Standardization Number TTAS.KO-12.0040.

2.2 Notation

For the sake of clarity and the parallelism with the previous work [13], we use exactly the same notation for HIGHT which is provided in Table 2.2. Throughout the paper, it is assumed that the rounds are numbered from zero and the leftmost bit is the most significant bit in a byte or a word.

Table 2.2: Notation

$\oplus$	Bitwise logical exclusive OR (XOR)
$\boxplus$	Addition modulo 2^8
$\lll$i	Left rotation by i bits
HIGHT-r	HIGHT reduced to r-rounds
e_j	A byte with zeros in all positions except bit j ($0 \leqslant j \leqslant 7$)
$e_{j,\sim}$	A byte that has zeros in bits 0 to $j-1$, a one in bit j and indeterminate values in bits $(j+1)$ to 7
$e_{\bar{j},\sim}$	A byte that has zeros in bits 0 to j and indeterminate values in bits $(j+1)$ to 7
?	An arbitrary byte
$X_{i,j}$	jth byte of state variable of round i of HIGHT, ($0 \leqslant j \leqslant 7$, $0 \leqslant i \leqslant 32$)
MK_i	ith Secret key byte of HIGHT
WK_i	ith Whitening key byte of HIGHT
SK_i	ith Subkey byte of HIGHT

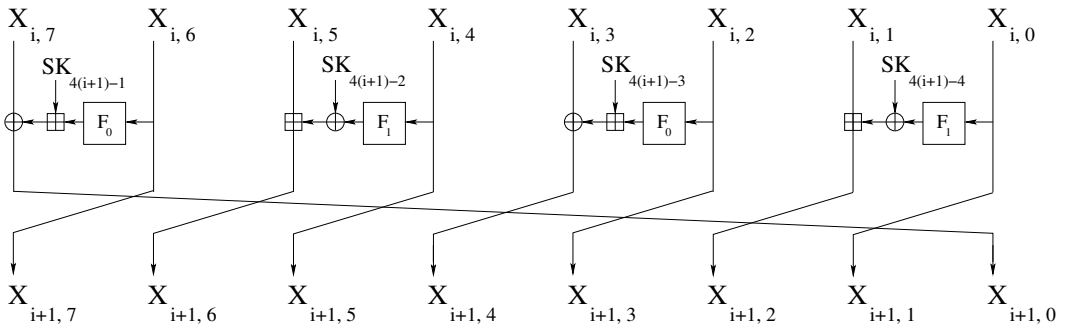

Figure 2.1: *i*th Round of HIGHT

2.3 Specifications

HIGHT is a 32-round block cipher with 64-bit block size and 128-bit user key that makes use of an unbalanced Feistel Network of 8 branches at each round. The encryption function starts with an Initial Transformation (IT) that is applied to plaintexts together with input whitening keys. After the execution of round function for 32 rounds, in order to obtain the ciphertexts, a Final Transformation (FT) is applied to the output of the last round together with output whitening keys. The round function, shown in Figure 2.1, operates on 8 bytes and uses simple operations such as bitwise XOR, addition modulo 2^8 and rotations.

Linear subround functions F_0 and F_1, shown in Figure 2.1, can be described as follows:

$$F_0(x) = (x \lll 1) \oplus (x \lll 2) \oplus (x \lll 7)$$
$$F_1(x) = (x \lll 3) \oplus (x \lll 4) \oplus (x \lll 6)$$

HIGHT works with a 128-bit secret key MK which is treated as 16 bytes, $(MK_{15}, \ldots, MK_0)$. The key schedule of HIGHT uses additional constants to avoid the self similarity in the key scheduling algorithm which prevents cipher from slide attacks [36]. Input-output whitening keys and round subkeys are obtained by permuting the 16 bytes of the original key and using addition with constants. The overall key scheduling can be described as:

For $i = 0$ to $i = 7$:
    If $0 \leq i \leq 3$, then $WK_i = MK_{i+12}$
    Else, $WK_i = MK_{i-4}$

For $i = 0$ to $i = 7$:
    For $j = 0$ to $j = 7$:
        $SK_{16i+j} = MK_{j-i \bmod 8} \boxplus \delta_{16i+j}$
    For $j = 0$ to $j = 7$:
        $SK_{16i+j+8} = MK_{(j-i \bmod 8)+8} \boxplus \delta_{16i+j+8}$

where δ_{16i+j} and $\delta_{16i+j+8}$ are public constants produced by a linear feedback shift register (LFSR). Generation of these constants by an LFSR enhances the randomness of the subkeys. Connection polynomial of this LFSR is $x^7 + x^3 + 1 \in \mathbb{Z}_2[x]$ and the initial state is 1011010_2. The period of $x^7 + x^3 + 1$ is $2^7 - 1 = 127$ since it is a primitive polynomial in $\mathbb{Z}_2[x]$. Hence only δ_0 and δ_{127} are the same.

Table 2.3 shows the relations between the original key and the subkey bytes and it will be extensively used in this study. Namely, each value in a row represents the obtained whitening and subkey bytes once the corresponding byte in the first column of the same row is known.

To reduce the memory requirements, only the values of MKs are kept and SKs and WKs are generated during encryption and decryption.

Table 2.3: Relations Between the Original Key and Whitening Keys and Subkeys

Original Key	Whitening Keys	Subkeys							
MK_{15}	WK_3	SK_{15}	SK_{24}	SK_{41}	SK_{58}	SK_{75}	SK_{92}	SK_{109}	SK_{126}
MK_{14}	WK_2	SK_{14}	SK_{31}	SK_{40}	SK_{57}	SK_{74}	SK_{91}	SK_{108}	SK_{125}
MK_{13}	WK_1	SK_{13}	SK_{30}	SK_{47}	SK_{56}	SK_{73}	SK_{90}	SK_{107}	SK_{124}
MK_{12}	WK_0	SK_{12}	SK_{29}	SK_{46}	SK_{63}	SK_{72}	SK_{89}	SK_{106}	SK_{123}
MK_{11}	-	SK_{11}	SK_{28}	SK_{45}	SK_{62}	SK_{79}	SK_{88}	SK_{105}	SK_{122}
MK_{10}	-	SK_{10}	SK_{27}	SK_{44}	SK_{61}	SK_{78}	SK_{95}	SK_{104}	SK_{121}
MK_9	-	SK_9	SK_{26}	SK_{43}	SK_{60}	SK_{77}	SK_{94}	SK_{111}	SK_{120}
MK_8	-	SK_8	SK_{25}	SK_{42}	SK_{59}	SK_{76}	SK_{93}	SK_{110}	SK_{127}
MK_7	-	SK_7	SK_{16}	SK_{33}	SK_{50}	SK_{67}	SK_{84}	SK_{101}	SK_{118}
MK_6	-	SK_6	SK_{23}	SK_{32}	SK_{49}	SK_{66}	SK_{83}	SK_{100}	SK_{117}
MK_5	-	SK_5	SK_{22}	SK_{39}	SK_{48}	SK_{65}	SK_{82}	SK_{99}	SK_{116}
MK_4	-	SK_4	SK_{21}	SK_{38}	SK_{55}	SK_{64}	SK_{81}	SK_{98}	SK_{115}
MK_3	WK_7	SK_3	SK_{20}	SK_{37}	SK_{54}	SK_{71}	SK_{80}	SK_{97}	SK_{114}
MK_2	WK_6	SK_2	SK_{19}	SK_{36}	SK_{53}	SK_{70}	SK_{87}	SK_{96}	SK_{113}
MK_1	WK_5	SK_1	SK_{18}	SK_{35}	SK_{52}	SK_{69}	SK_{86}	SK_{103}	SK_{112}
MK_0	WK_4	SK_0	SK_{17}	SK_{34}	SK_{51}	SK_{68}	SK_{85}	SK_{102}	SK_{119}

Let $X_i = (X_{i,7}, \ldots, X_{i,0})$ be the input of the ith round and $X_{i+1} = (X_{i+1,7}, \ldots, X_{i+1,0})$ denote its output. If we denote plaintexts by $P = (P_7, .., P_0)$ and ciphertexts by $C = (C_7, .., C_0)$, then we can describe the encryption function as follows:

1. Initial Transformation :

$X_{0,7} = P_7$
$X_{0,6} = P_6 \oplus WK_3$
$X_{0,5} = P_5$
$X_{0,4} = P_4 \boxplus WK_2$
$X_{0,3} = P_3$
$X_{0,2} = P_2 \oplus WK_1$
$X_{0,1} = P_1$
$X_{0,0} = P_0 \boxplus WK_0$

2. Round Function, for $i = 1$ to 32:

$x_{i,0} = X_{i-1,7} \oplus (F_0(X_{i-1,6}) \boxplus SK_{4i-1})$
$x_{i,1} = X_{i-1,0}$
$x_{i,2} = X_{i-1,1} \boxplus (F_1(X_{i-1,0}) \oplus SK_{4i-4})$
$x_{i,3} = X_{i-1,2}$
$x_{i,4} = X_{i-1,3} \oplus (F_0(X_{i-1,2}) \boxplus SK_{4i-3})$
$x_{i,5} = X_{i-1,4}$
$x_{i,6} = X_{i-1,5} \boxplus (F_1(X_{i-1,4}) \oplus SK_{4i-2})$
$x_{i,7} = X_{i-1,6}$

3. Final Transformation :

$C_7 = X_{32,0}$
$C_6 = X_{32,7} \oplus WK_7$
$C_5 = X_{32,6}$
$C_4 = X_{32,5} \boxplus WK_6$
$C_3 = X_{32,4}$
$C_2 = X_{32,3} \oplus WK_5$
$C_1 = X_{32,2}$
$C_0 = X_{32,1} \boxplus WK_4$

2.4 Previous Impossible Differential Attacks on HIGHT

The security of HIGHT is investigated in [2] by showing resistance against known attacks such as differential, linear, truncated differential, boomerang [38], rectangle [39], impossible differential and related-key variants of these attacks. In [2], the safety margin was shown to be 13 rounds, as the best attack, which is a related-key boomerang attack, covers 19 rounds where the impossible differential attack covers 18 rounds of HIGHT.

Recently, three new attacks are proposed by Lu [12, 13] on reduced round HIGHT which are 25-round impossible differential, 26-round related-key rectangle and 28-round related-key impossible differential attacks. Last of these attacks was the best attack on HIGHT so

far which reduced the safety margin of HIGHT from 13 rounds to 4 rounds. In all of these attacks, the following two commonly known properties of XOR and modular addition is used.

Property 1: The XOR operation preserves differences. That is, if $a \oplus b = c$ then $(a \oplus d) \oplus (b \oplus d) = c$.

Property 2: The modular addition operation preserves the first least significant difference but other differences may not be preserved. That is, if $a \oplus b = c$ where the least significant i bits of c is 0 and i-th bit is 1, then $(a \boxplus d) \oplus (b \boxplus d) = e_{i,\sim}$.

2.4.1 Attack on 18-round HIGHT

In the proposal of HIGHT, designers claim that they investigated all of the possible input differences and found a 14-round impossible differential characteristic which is given as $\alpha \rightarrow \beta \neq \gamma \leftarrow \delta$ where $\alpha = (e_7, e_{0,3,5,6,7}, 0, 0, 0, 0, 0, 0)$, $\beta = (e_{0,\sim}, ?, ?, ?, ?, ?, ?, ?)$, $\gamma = (0, 0, ?, ?, ?, ?, ?, ?)$ and $\delta = (0, ?, ?, ?, 0, 0, 0, 0)$. Details of this characteristic can be found in Table 2.4 where the bytes which cause the differentials to miss in the middle are denoted with a gray background.

Although the attack is not explicitly given in the proposal, authors claim that an impossible differential cryptanalysis of 18-round HIGHT can be done by using this characteristic and the attack requires $2^{46.8}$ chosen-plaintexts and $2^{109.2}$ HIGHT-18 encryptions.

2.4.2 Attack on 25-round HIGHT

In [12], a 16-round impossible differential characteristic is given as $\alpha \rightarrow \beta \neq \gamma \leftarrow \delta$ where $\alpha = (e_{1,\sim}, 0, 0, 0, 0, 0, 0, 0)$, $\beta = (e_{1,\sim}, ?, ?, ?, ?, ?, ?, ?)$, $\gamma = (e_{0,\sim}, 0, ?, ?, ?, ?, ?, ?)$ and $\delta = (e_{0,3,5,6,7}, 0, 0, 0, 0, 0, 0, e_7)$. We give this characteristic in detail in Table 2.5 where the bytes which cause the differentials to miss in the middle are denoted with a gray background.

This characteristic can be used to attack 25-round HIGHT without the initial transformation and recover the whole secret key with 2^{60} chosen-plaintexts and $2^{126.78}$ HIGHT-25 encryptions. The attack procedure is similar to our 26-round impossible differential attack that we are going to describe at Chapter 3 and hence it is not given here.

Table 2.4: 13-Round Impossible Differential Characteristic

	$\Delta X_{i,7}$	$\Delta X_{i,6}$	$\Delta X_{i,5}$	$\Delta X_{i,4}$	$\Delta X_{i,3}$	$\Delta X_{i,2}$	$\Delta X_{i,1}$	$\Delta X_{i,0}$
1	e_7	$e_{0,3,5,6,7}$	0	0	0	0	0	0
2	$e_{0,3,5,6,7}$	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	$e_{0,3,5,6,7}$
4	0	0	0	0	0	?	$e_{0,3,5,6,7}$	0
5	0	0	0	?	?	$e_{0,\sim}$	0	0
6	0	?	?	?	$e_{0,\sim}$	0	0	0
7	?	?	?	$e_{0,\sim}$	0	0	0	?
8	?	?	$e_{0,\sim}$	0	0	?	?	?
9	?	$e_{0,\sim}$	0	?	?	?	?	?
10	$e_{0,\sim}$	?	?	?	?	?	?	?
10	0	0	?	?	?	?	?	?
11	0	0	?	?	?	?	?	0
12	0	0	?	?	?	?	0	0
13	0	0	?	?	?	0	0	0
14	0	?	?	?	0	0	0	0

Table 2.5: 16-Round Impossible Differential Characteristic

	$\Delta X_{i,7}$	$\Delta X_{i,6}$	$\Delta X_{i,5}$	$\Delta X_{i,4}$	$\Delta X_{i,3}$	$\Delta X_{i,2}$	$\Delta X_{i,1}$	$\Delta X_{i,0}$
1	$e_{1,\sim}$	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	$e_{1,\sim}$
3	0	0	0	0	0	?	$e_{1,\sim}$	0
4	0	0	0	?	?	$e_{1,\sim}$	0	0
5	0	?	?	?	$e_{1,\sim}$	0	0	0
6	?	?	?	$e_{1,\sim}$	0	0	0	?
7	?	?	$e_{1,\sim}$	0	0	?	?	?
8	?	$e_{1,\sim}$	0	?	?	?	?	?
9	$e_{1,\sim}$	?	?	?	?	?	?	?
9	$e_{0,\sim}$	0	?	?	?	?	?	?
10	0	0	?	?	?	?	?	$e_{0,\sim}$
11	0	0	?	?	?	?	$e_{0,\sim}$	0
12	0	0	?	?	?	$e_{0,\sim}$	0	0
13	0	0	?	?	$e_{0,\sim}$	0	0	0
14	0	0	?	$e_{0,\sim}$	0	0	0	0
15	0	0	$e_{0,\sim}$	0	0	0	0	0
16	0	$e_{0,3,5,6,7}$	0	0	0	0	0	0
17	$e_{0,3,5,6,7}$	0	0	0	0	0	0	e_7

2.5 Previous Related-key Impossible Differential Attacks on HIGHT

There are no related-key impossible differential attacks in the proposal of HIGHT and authors are "convinced that the key schedule and round function of HIGHT makes related-key attacks difficult" [2]. The best related-key attack they could find is a 19-round related-key boomerang attack. However, in [12, 13], Lu presented a 26-round related-key boomerang attack and a 28-round related-key impossible differential attack on HIGHT.

2.5.1 Attack on 28-round HIGHT

In [12], a 19-round related-key impossible differential characteristic which starts at round 7 and ends at round 25 is given as $(e_7, 0, 0, 0, 0, 0, 0, 0) \nrightarrow (0, 0, 0, 0, 0, 0, 0, e_{1,\sim})$ where the key difference $(\Delta MK_{15}, \Delta MK_{14}, \ldots, \Delta MK_0)$ is $(0, 0, 0, 0, 0, e_7, 0, \ldots, 0)$. We give this characteristic in detail in Table 2.6 where the bytes which cause the differentials to miss in the middle are denoted with a light gray background and the subkeys which are affected by the key difference are denoted with a dark gray background.

This characteristic can be used to attack HIGHT-28 without the initial transformation, which starts at round 2 and end at round 29. This attack requires 2^{60} chosen plaintexts and $2^{125.54}$ HIGHT-28 encryptions. The attack procedure is similar to our 31-round related-key impossible differential attack that we are going to describe at Chapter 4 and hence it is not given here.

Table 2.6: 19-Round Related-key Impossible Differential Characteristic, $\Delta MK_{10} = e_7$

	$\Delta X_{i,7}$	$\Delta X_{i,6}$	$\Delta X_{i,5}$	$\Delta X_{i,4}$	$\Delta X_{i,3}$	$\Delta X_{i,2}$	$\Delta X_{i,1}$	$\Delta X_{i,0}$	Subkeys			
ΔX_6	e_7	0	0	0	0	0	0	0	SK_{27}	SK_{26}	SK_{25}	SK_{24}
ΔX_7	0	0	0	0	0	0	0	0	SK_{31}	SK_{30}	SK_{29}	SK_{28}
ΔX_8	0	0	0	0	0	0	0	0	SK_{35}	SK_{34}	SK_{33}	SK_{32}
ΔX_9	0	0	0	0	0	0	0	0	SK_{39}	SK_{38}	SK_{37}	SK_{36}
ΔX_{10}	0	0	0	0	0	0	0	0	SK_{43}	SK_{42}	SK_{41}	SK_{40}
ΔX_{11}	0	0	0	0	0	0	0	0	SK_{47}	SK_{46}	SK_{45}	SK_{44}
ΔX_{12}	0	0	0	0	0	e_7	0	0	SK_{51}	SK_{50}	SK_{49}	SK_{48}
ΔX_{13}	0	0	0	$e_{0,\sim}$	e_7	0	0	0	SK_{55}	SK_{54}	SK_{53}	SK_{52}
ΔX_{14}	0	?	$e_{0,\sim}$	e_7	0	0	0	0	SK_{59}	SK_{58}	SK_{57}	SK_{56}
ΔX_{15}	?	$e_{0,\sim}$	e_7	0	0	0	0	?	SK_{63}	SK_{62}	SK_{61}	SK_{60}
ΔX_{16}	$e_{0,\sim}$	e_7	0	e_7	0	?	?	?	SK_{67}	SK_{66}	SK_{65}	SK_{64}
ΔX_{17}	e_7	$e_{2,\sim}$	e_7	?	?	?	?	$e_{\tilde{0},\sim}$	SK_{71}	SK_{70}	SK_{69}	SK_{68}
ΔX_{18}	$e_{2,\sim}$	e_7	?	?	?	?	$e_{\tilde{0},\sim}$	?	SK_{75}	SK_{74}	SK_{73}	SK_{72}
ΔX_{18}	?	?	?	?	?	?	$e_{0,\sim}$	0	SK_{75}	SK_{74}	SK_{73}	SK_{72}
ΔX_{19}	?	?	?	?	?	$e_{0,\sim}$	0	0	SK_{79}	SK_{78}	SK_{77}	SK_{76}
ΔX_{20}	?	?	?	?	$e_{0,\sim}$	0	0	0	SK_{83}	SK_{82}	SK_{81}	SK_{80}
ΔX_{21}	?	?	?	$e_{0,\sim}$	0	0	0	0	SK_{87}	SK_{86}	SK_{85}	SK_{84}
ΔX_{22}	?	?	$e_{0,\sim}$	0	0	0	0	0	SK_{91}	SK_{90}	SK_{89}	SK_{88}
ΔX_{23}	?	$e_{0,\sim}$	0	0	0	0	0	0	SK_{95}	SK_{94}	SK_{93}	SK_{92}
ΔX_{24}	$e_{0,\sim}$	0	0	0	0	0	0	0	SK_{99}	SK_{98}	SK_{97}	SK_{96}
ΔX_{25}	0	0	0	0	0	0	0	$e_{0,\sim}$	SK_{103}	SK_{102}	SK_{101}	SK_{100}

CHAPTER 3

IMPOSSIBLE DIFFERENTIAL CRYPTANALYSIS

In this chapter we introduce our impossible differential attack on HIGHT-26 which uses a 16-round impossible differential characteristic and covers the rounds 0-25. The input whitening is excluded in our attack as done in [13]. To our knowledge, this is the best attack on HIGHT among the attacks that does not use related-keys.

3.1 17-round Impossible Differential Characteristic

Because of the structure of HIGHT, one might observe that a given difference becomes undetermined after at most 10 rounds of encryption and there are only a few number of differences that does not become undetermined after 9 rounds of encryption. One example is the difference $(e_7, e_{0,3,5,6,7}, 0, 0, 0, 0, 0, 0)$ which is shown in detail in Table 2.4.

Therefore we can use two 9-round truncated differentials which do not meet in the middle and obtain the theoretical highest round impossible differential characteristic for the miss in the middle technique which is of 18 rounds. However, we checked every 9-round differential (which holds with probability 1) and observed that when two of them is combined, they do not miss in the middle.

Hence theoretical 18-round impossible differential characteristic is not possible in practice and Lu [12] uses a 16-round impossible differential characteristic (which is given in detail in Table 2.4) in his 25-round impossible differential attack. Thus, we tried to find a 17-round impossible differential characteristic in order to attack more rounds of HIGHT and obtained a 17-round impossible differential characteristic which is shown in detail in Table 3.1 where the bytes which cause the differentials to miss in the middle are denoted with a gray background.

Table 3.1: 17-Round Impossible Differential Characteristic

	$\Delta X_{i,7}$	$\Delta X_{i,6}$	$\Delta X_{i,5}$	$\Delta X_{i,4}$	$\Delta X_{i,3}$	$\Delta X_{i,2}$	$\Delta X_{i,1}$	$\Delta X_{i,0}$
1	e_7	$e_{0,3,5,6,7}$	0	0	0	0	0	0
2	$e_{0,3,5,6,7}$	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	$e_{0,3,5,6,7}$
4	0	0	0	0	0	?	$e_{0,3,5,6,7}$	0
5	0	0	0	?	?	$e_{0,\sim}$	0	0
6	0	?	?	?	$e_{0,\sim}$	0	0	0
7	?	?	?	$e_{0,\sim}$	0	0	0	?
8	?	?	$e_{0,\sim}$	0	0	?	?	?
9	?	$e_{0,\sim}$	0	?	?	?	?	?
10	$e_{0,\sim}$	?	?	?	?	?	?	?
10	$e_{\bar{0},\sim}$	e_7	?	?	?	?	?	?
11	e_7	0	?	?	?	?	?	$e_{0,\sim}$
12	0	0	?	?	?	?	$e_{0,\sim}$	e_7
13	0	0	?	?	?	$e_{0,\sim}$	e_7	0
14	0	0	?	?	$e_{0,\sim}$	e_7	0	0
15	0	0	?	$e_{2,\sim}$	e_7	0	0	0
16	0	0	$e_{2,\sim}$	e_7	0	0	0	0
17	0	0	e_7	0	0	0	0	0
18	0	e_7	0	0	0	0	0	0

Here we explain why the impossible differential characteristic given in Table 3.1 holds with probability 1. We start from the 1st row of Table 3.1 and reach to 10th row by noting the behaviour of the difference. Since $F_0(e_{0,3,5,6,7}) = e_7$, it cancels $\Delta X_{1,7}$, so $\Delta X_{2,0}$ becomes 0. At 5th row, $e_{0,3,5,6,7}$ becomes $e_{0,\sim}$ because of the modular addition and Property 2. Other steps follow similarly. Hence with the given difference we conclude that after 9 rounds of encryption, the least significant bit of $\Delta X_{10,7}$ has to be 1.

Now we start from the 18th row of the Table 3.1 and note the behaviour of the given difference by going upwards, in other words, by decrypting. When we go upwards from 17th row to 16th row, the e_7 difference at $\Delta X_{17,5}$ goes to $\Delta X_{16,4}$. We know that $F_1(e_7) = e_{2,3,5}$ which becomes $e_{2,\sim}$ after modular addition but $\Delta X_{17,6}$ is 0, so $\Delta X_{16,5}$ must be $e_{2,\sim}$ to cancel out $F_1(e_7)$. Other steps follow similarly. Hence with the given difference we conclude that after 8 rounds of decryption, the least significant bit of $\Delta X_{10,7}$ has to be 0.

Since both of the differentials hold with probability 1, we conclude that if two messages have the difference $(e_7, e_{0,3,5,6,7}, 0, 0, 0, 0, 0, 0)$, after 17 rounds of encryption, the difference between them can not be $(0, e_7, 0, 0, 0, 0, 0, 0)$. In other words, such a characteristic is impos-

sible to hold.

With this impossible differential characteristic, we can attack 26 rounds of HIGHT. However, the time complexity of this attack is very close to the exhaustive search. This is due to the fact that the above impossible differential characteristic requires too many filtering conditions to hold and these conditions eliminate too many pairs and at the end, remaining pairs are not enough to eliminate most of the wrong keys. We overcome this problem by eliminating the first row of the Table 3.1 and replacing $e_{0,3,5,6,7}$ by $e_{0,\sim}$. Hence our characteristic reduces to 16 rounds but as we will see in the following sections the complexity of the attack becomes much lower with this characteristic.

The 16-round impossible differential characteristic that we are going to use can be shown as

$$(e_{0,\sim}, 0, 0, 0, 0, 0, 0, 0) \nrightarrow (0, e_7, 0, 0, 0, 0, 0, 0). \tag{3.1}$$

Because of the symmetry in the round function of HIGHT, another 16-round impossible differential can be obtained as

$$(0, 0, 0, 0, e_{0,\sim}, 0, 0, 0) \nrightarrow (0, 0, 0, 0, 0, e_7, 0, 0). \tag{3.2}$$

3.2 26-round Path

Now we have constructed our 16-round impossible differential characteristic and we will use it to attack 26-rounds of HIGHT. We extend our 16-round impossible differential characteristic to a 26-round path by adding 5 rounds to above of the characteristic and 5 rounds to below of the characteristic. This path is shown in Table 3.2 where the bytes which cause the differentials to miss in the middle are denoted with a light gray background and the subkeys and whitening keys that are guessed during the attack process are denoted with a dark gray background.

By using this 26-round path, we will eliminate a key if a plaintext-ciphertext pair satisfies the 16-round impossible differential characteristic under that key. In order to check whether the characteristic is satisfied or not, we must partially encrypt the plaintext pairs and partially decrypt the corresponding ciphertext pairs. Hence, during these processes we need to guess corresponding subkey values and this is done by guessing the corresponding secret key bytes.

We want to perform an attack which requires minimum number of secret key byte guesses because it directly effects the time and memory complexity of the attack.

Since different subkeys are used in each round, which 26 rounds of HIGHT we are going to attack becomes important. By writing a simple programming code we checked that if the 26-round path covers the first 26 rounds of HIGHT, we need to guess 112 bits of the key, namely $(MK_{15}, MK_{14}, MK11, \ldots, MK_0)$ and this is one of the lowest possible. The 26-round path is given in Table 3.2. This attack covers the rounds 0-25 and excludes the input whitening as done in [13].

3.3 Data Collection and Memory

1. We choose 2^{13} structures of 2^{48} plaintexts P^i each where the bytes $(1, 0)$ have fixed values, bytes $(7, 6, 5, 4, 3)$ and most significant 7 bits of the byte (2) take all possible values.

    - Let 2^{47} of the plaintexts have the bit 0 as their least significant bit of byte (2) and remaining 2^{47} plaintexts have 1 in that place. If we construct pairs of plaintext from these two sets of plaintexts, such a structure of plaintexts propose exactly $2^{47} \cdot 2^{47} = 2^{94}$ plaintext pairs and every pair will have the difference (?, ?, ?, ?, ?, $e_{0,\sim}$, 0, 0). Since we have 2^{13} structures, we get $2^{94} \cdot 2^{13} = 2^{107}$ pairs in total.

2. We obtain all the ciphertexts C^i of the plaintexts P^i and choose only the ciphertext pairs satisfying the difference $(?, ?, ?, ?, e_{\tilde{0},\sim}, e_7, 0, 0)$.

    - This step can be done by inserting all the ciphertexts into a hash table indexed by expected inactive bits and choosing the colliding pairs which satisfy the required difference. There is 25-bit filtering condition over the ciphertext pairs. Therefore, $2^{107}/2^{25} = 2^{82}$ pairs remain.

3. This attack will be on 112 bits of the secret key, namely (MK_{15}, MK_{14}, $MK11$, ... , MK_0) and since we are going to eliminate wrong keys, we have to store them. One trivial way of doing this is by forming an array of bits with size 2^{112} and initialize them to 0. The r-th element of the array represents the key whose integer value is r. When a key is eliminated, we will convert its corresponding value in the array to 1.

Table 3.2: 26-Round Impossible Differential Path

	$\Delta X_{i,7}$	$\Delta X_{i,6}$	$\Delta X_{i,5}$	$\Delta X_{i,4}$	$\Delta X_{i,3}$	$\Delta X_{i,2}$	$\Delta X_{i,1}$	$\Delta X_{i,0}$	Subkeys			
ΔX_0	?	?	?	?	?	$e_{0,\sim}$	0	0	SK_3	SK_2	SK_1	SK_0
ΔX_1	?	?	?	?	$e_{0,\sim}$	0	0	0	SK_7	SK_6	SK_5	SK_4
ΔX_2	?	?	?	$e_{0,\sim}$	0	0	0	0	SK_{11}	SK_{10}	SK_9	SK_8
ΔX_3	?	?	$e_{0,\sim}$	0	0	0	0	0	SK_{15}	SK_{14}	SK_{13}	SK_{12}
ΔX_4	?	$e_{0,\sim}$	0	0	0	0	0	0	SK_{19}	SK_{18}	SK_{17}	SK_{16}
ΔX_5	$e_{0,\sim}$	0	0	0	0	0	0	0	SK_{23}	SK_{22}	SK_{21}	SK_{20}
ΔX_6	0	0	0	0	0	0	0	$e_{0,\sim}$	SK_{27}	SK_{26}	SK_{25}	SK_{24}
ΔX_7	0	0	0	0	0	?	$e_{0,\sim}$	0	SK_{31}	SK_{30}	SK_{29}	SK_{28}
ΔX_8	0	0	0	?	?	$e_{0,\sim}$	0	0	SK_{35}	SK_{34}	SK_{33}	SK_{32}
ΔX_9	0	?	?	?	$e_{0,\sim}$	0	0	0	SK_{39}	SK_{38}	SK_{37}	SK_{36}
ΔX_{10}	?	?	?	$e_{0,\sim}$	0	0	0	?	SK_{43}	SK_{42}	SK_{41}	SK_{40}
ΔX_{11}	?	?	$e_{0,\sim}$	0	0	?	?	?	SK_{47}	SK_{46}	SK_{45}	SK_{44}
ΔX_{12}	?	$e_{0,\sim}$	0	?	?	?	?	?	SK_{51}	SK_{50}	SK_{49}	SK_{48}
ΔX_{13}	$e_{0,\sim}$	?	?	?	?	?	?	?	SK_{55}	SK_{54}	SK_{53}	SK_{52}
ΔX_{13}	$e_{\bar{0},\sim}$	e_7	?	?	?	?	?	?	SK_{55}	SK_{54}	SK_{53}	SK_{52}
ΔX_{14}	e_7	0	?	?	?	?	?	$e_{0,\sim}$	SK_{59}	SK_{58}	SK_{57}	SK_{56}
ΔX_{15}	0	0	?	?	?	?	$e_{0,\sim}$	e_7	SK_{63}	SK_{62}	SK_{61}	SK_{60}
ΔX_{16}	0	0	?	?	?	$e_{0,\sim}$	e_7	0	SK_{67}	SK_{66}	SK_{65}	SK_{64}
ΔX_{17}	0	0	?	?	$e_{0,\sim}$	e_7	0	0	SK_{71}	SK_{70}	SK_{69}	SK_{68}
ΔX_{18}	0	0	?	$e_{2,\sim}$	e_7	0	0	0	SK_{75}	SK_{74}	SK_{73}	SK_{72}
ΔX_{19}	0	0	$e_{2,\sim}$	e_7	0	0	0	0	SK_{79}	SK_{78}	SK_{77}	SK_{76}
ΔX_{20}	0	0	e_7	0	0	0	0	0	SK_{83}	SK_{82}	SK_{81}	SK_{80}
ΔX_{21}	0	e_7	0	0	0	0	0	0	SK_{87}	SK_{86}	SK_{85}	SK_{84}
ΔX_{22}	e_7	0	0	0	0	0	0	$e_{0,\sim}$	SK_{91}	SK_{90}	SK_{89}	SK_{88}
ΔX_{23}	0	0	0	0	0	?	$e_{0,\sim}$	e_7	SK_{95}	SK_{94}	SK_{93}	SK_{92}
ΔX_{24}	0	0	0	?	?	$e_{0,\sim}$	e_7	0	SK_{99}	SK_{98}	SK_{97}	SK_{96}
ΔX_{25}	0	?	?	?	$e_{0,\sim}$	e_7	0	0	SK_{103}	SK_{102}	SK_{101}	SK_{100}
ΔX_{26}	?	?	?	$e_{\bar{0},\sim}$	e_7	0	0	?	WK_7	WK_6	WK_5	WK_4
FT	?	?	?	?	$e_{\bar{0},\sim}$	e_7	0	0				

3.4 Impossible Differential Attack on HIGHT-26

Attack procedure is to take a pair and perform partial encryptions and decryptions by guessing the necessary key bytes and eliminating them if they satisfy the impossible differential characteristic. The following steps are applied to every pair that we obtained in the data collection part.

1. We guess MK_3 and get SK_3 (actually by guessing MK_3, we get WK_7, SK_3, SK_{20}, SK_{37}, SK_{54}, SK_{71}, SK_{80}, SK_{97} and SK_{114}, which can be checked from Table 2.3). Since we know the values of the plaintexts, we obtain the bytes $(7, 0)$ of X_1 by partially encrypting every plaintexts' bytes $(7, 6)$ of X_0 with SK_3. In order to satisfy the impossible differential characteristic, the obtained values of the pairs must have the difference $(?, 0)$. We keep the pairs satisfying this condition. Since the difference $(?, 0)$ contains 8-bit filtering condition, 2^8 of the pairs are eliminated in this step. Hence $2^{82}/2^8 = 2^{74}$ pairs remain.

2. We guess MK_1 and obtain the bytes $(7, 6)$ of X_{25} by partially decrypting the bytes $(7, 0)$ of X_{26} with WK_7 and SK_{103} for all remaining pairs. We keep the pairs having the difference $(0, ?)$ at the bytes $(7, 6)$ of X_{25}. It contains 8-bit filtering condition so 2^{66} pairs remain.

3. We guess MK_2 and obtain the bytes $(6, 5)$ of X_1 by partially encrypting the bytes $(5, 4)$ of X_0 with SK_2 for all remaining pairs. There are no filtering conditions in this step since the required difference is $(?, ?)$.

4. We guess MK_7 and obtain the bytes $(7, 0)$ of X_2 by partially encrypting the bytes $(7, 6)$ of X_1 with SK_7 for all remaining pairs. We keep the pairs having the difference $(?, 0)$ at the bytes $(7, 0)$ of X_2. It contains 8-bit filtering condition so 2^{58} pairs remain.

5. We guess MK_0 and obtain the bytes $(5, 4)$ of X_{25} by partially decrypting the bytes $(6, 5)$ of X_{26} with WK_6 and SK_{102} for all remaining pairs. There are no filtering conditions in this step.

6. We guess MK_4 and obtain the bytes $(5, 4)$ of X_{24} by partially decrypting the bytes $(6, 5)$ of X_{25} with SK_{98} for all remaining pairs. We keep the pairs having the difference $(0, ?)$ at the bytes $(5, 4)$ of X_{24}. It contains 8-bit filtering condition so 2^{50} pairs remain.

7. We obtain the bytes $(3,2)$ of X_{25} by partially decrypting the bytes $(4,3)$ of X_{26} with WK_5 and SK_{101} for all remaining pairs. Since we already guessed the secret key bytes MK_1 and MK_7, we know the values of WK_5 and SK_{101}. There are no filtering conditions in this step.

8. We obtain the bytes $(3,2)$ of X_{24} by partially decrypting the bytes $(4,3)$ of X_{25} with SK_{97} for all remaining pairs. Since we already guessed the secret key byte MK_3, we know the value of SK_{97}. There are no filtering conditions in this step.

9. We guess MK_8 and obtain the bytes $(3,2)$ of X_{23} by partially decrypting $(4,3)$ of X_{24} with SK_{93} for all remaining pairs. We keep the pairs having the difference $(0,?)$ at the bytes $(3,2)$ of X_{23}. It contains 8-bit filtering condition so 2^{42} pairs remain.

10. We obtain the bytes $(4,3)$ of X_1 by partially encrypting $(3,2)$ of X_0 with SK_1 for all remaining pairs. Since we already guessed the secret key byte MK_1, we know the value of SK_1. There are no filtering conditions in this step.

11. We guess MK_6 and obtain the bytes $(6,5)$ of X_2 by partially encrypting $(5,4)$ of X_1 with SK_6 for all remaining pairs. There are no filtering conditions in this step.

12. We guess MK_{11} and obtain the bytes $(7,0)$ of X_3 by partially encrypting $(7,6)$ of X_2 with SK_{11} for all remaining pairs. We keep the pairs having the difference $(?,0)$ at the bytes $(7,0)$ of X_3. It contains 8-bit filtering condition so 2^{34} pairs remain.

13. We obtain the bytes $(1,0)$ of X_{25} by partially decrypting the bytes $(2,1)$ of X_{26} with WK_4 and SK_{100} for all remaining pairs. Since we already guessed the secret key bytes MK_0 and MK_6, we know the values of WK_4 and SK_{100}. There are no filtering conditions in this step.

14. We obtain the bytes $(1,0)$ of X_{24} by partially decrypting the bytes $(2,1)$ of X_{25} with SK_{96} for all remaining pairs. Since we already guessed the secret key byte MK_2, we know the value of SK_{96}. There are no filtering conditions in this step.

15. We guess MK_{15} and obtain the bytes $(1,0)$ of X_{23} by partially decrypting $(2,1)$ of X_{24} with SK_{92} for all remaining pairs. There are no filtering conditions in this step.

16. We obtain the bytes $(1,0)$ of X_{22} by partially decrypting the bytes $(2,1)$ of X_{23} with SK_{88} for all remaining pairs. Since we already guessed the secret key byte MK_{11}, we

know the value of SK_{88}. We keep the pairs having the difference $(0, e_{0,\sim})$ at the bytes $(1, 0)$ of X_{22}. It contains 8-bit filtering condition so 2^{26} pairs remain.

17. We obtain the bytes $(2, 1)$ of X_1 by partially encrypting $(1, 0)$ of X_0 with SK_0 for all remaining pairs. Since we already guessed the secret key byte MK_0, we know the value of SK_0. There are no filtering conditions in this step.

18. We guess MK_5 and obtain the bytes $(4, 3)$ of X_2 by partially encrypting $(3, 2)$ of X_1 with SK_5 for all remaining pairs. There are no filtering conditions in this step.

19. We guess MK_{10} and obtain the bytes $(6, 5)$ of X_3 by partially encrypting $(5, 4)$ of X_2 with SK_{10} for all remaining pairs. There are no filtering conditions in this step.

20. We obtain the bytes $(7, 0)$ of X_4 by partially encrypting $(7, 6)$ of X_3 with SK_{15} for all remaining pairs. Since we already guessed the secret key byte MK_{15}, we know the value of SK_{15}. We keep the pairs having the difference $(?, 0)$ at the bytes $(7, 0)$ of X_4. It contains 8-bit filtering condition so 2^{18} pairs remain.

21. We obtain the bytes $(7, 6)$ of X_{24} by partially decrypting the bytes $(7, 0)$ of X_{25} with SK_{99} for all remaining pairs. Since we already guessed the secret key byte MK_5, we know the value of SK_{99}. There are no filtering conditions in this step.

22. We obtain the bytes $(7, 6)$ of X_{23} by partially decrypting the bytes $(7, 0)$ of X_{24} with SK_{95} for all remaining pairs. Since we already guessed the secret key byte MK_{10}, we know the value of SK_{95}. There are no filtering conditions in this step.

23. We guess MK_{14} and obtain the bytes $(7, 6)$ of X_{22} by partially decrypting $(7, 0)$ of X_{23} with SK_{91} for all remaining pairs. There are no filtering conditions in this step.

24. We obtain the bytes $(7, 6)$ of X_{21} by partially decrypting the bytes $(7, 0)$ of X_{22} with SK_{87} for all remaining pairs. Since we already guessed the secret key byte MK_2, we know the value of SK_{87}. We keep the pairs having the difference $(0, e_7)$ at the bytes $(7, 6)$ of X_{21}. It contains 7-bit filtering condition because $F_0(e_7)$ becomes $e_{0,\sim}$ after modular addition and byte (7) of X_{21} becomes $e_{0,\sim} \oplus e_{0,\sim} = e_{\bar{0},\sim}$. So the least significant bit is always 0. Thus the filtering conditions are on the remaining 7 bits of byte (7) of X_{21}. Hence 2^{11} pairs remain.

25. We obtain the bytes $(2, 1)$ of X_2 by partially encrypting $(1, 0)$ of X_1 with SK_4 for all remaining pairs. Since we already guessed the secret key byte MK_4, we know the value of SK_4. There are no filtering conditions in this step.

26. We guess MK_9 and obtain the bytes $(4, 3)$ of X_3 by partially encrypting $(3, 2)$ of X_2 with SK_9 for all remaining pairs. There are no filtering conditions in this step.

27. We obtain the bytes $(6, 5)$ of X_4 by partially encrypting $(5, 4)$ of X_3 with SK_{14} for all remaining pairs. Since we already guessed the secret key byte MK_{14}, we know the value of SK_{14}. There are no filtering conditions in this step.

28. We obtain the bytes $(7, 0)$ of X_5 by partially encrypting $(7, 6)$ of X_4 with SK_{19} for all remaining pairs. Since we already guessed the secret key byte MK_2, we know the value of SK_{19}. We check if the obtained difference is $(e_{0,\sim}, 0)$ at the bytes $(7, 0)$ of X_5. If a pair satisfies this condition, we eliminate the corresponding key. Since there is an 8-bit condition, every pair eliminates 2^{-8} of the keys. Therefore after the first pair, there remain $2^{112} - 2^{104} = 2^{112} \cdot (1 - 2^{-8})$ keys. After the second pair, it is expected to have $2^{112} \cdot (1 - 2^{-8}) - 2^{112} \cdot (1 - 2^{-8}) \cdot 2^{-8} = 2^{112} \cdot (1 - 2^{-8})^2$ many keys. Following that manner, after the last pair, we have $2^{112} \cdot (1 - 2^{-8})^{2^{11}} \approx 2^{100.46}$ remaining keys.

29. Now we reduced the number of possible keys from 2^{112} to $2^{100.46}$. For all these remaining keys, we guess the remaining 16 bits of the key, that is MK_{13} and MK_{12}, and exhaustively search for the correct one. Hence we check $2^{100.46+16} = 2^{116.46}$ keys in this step. This is done by encrypting one plaintext with the guessed key and checking whether the corresponding ciphertext is obtained. Sometimes with a wrong key, we might observe that a plaintext is encrypted to the corresponding ciphertext. Such a situation is called a *false alarm* and its probability is 2^{-64} since the block size of HIGHT is 64 bits. To avoid false alarms, if a correct ciphertext is obtained with a key, an another plaintext-ciphertext pair should be checked with the same key. If the correct ciphertext is obtained again, we conclude that it is the correct secret key because the probability of obtaining the correct ciphertexts two times with a wrong key is $(2^{-64})^2 = 2^{-128}$ and expected number of wrong keys is $2^{-128} \cdot 2^{116.46} = 2^{-11.54}$.

These steps are summarized in Table 3.3.

Table 3.3: 26-Round impossible differential attack

	Guess Key Byte	Use	Obtain	Check Difference	Condition (In terms of bits)	Remaining Pairs
1	MK_3	SK_3	$(7,0)$ of X_1	$(?,0)$	8	2^{74}
2	MK_1	WK_7, SK_{103}	$(7,6)$ of X_{25}	$(0,?)$	8	2^{66}
3	MK_2	SK_2	$(6,5)$ of X_1	-	-	2^{66}
4	MK_7	SK_7	$(7,0)$ of X_2	$(?,0)$	8	2^{58}
5	MK_0	WK_6, SK_{102}	$(5,4)$ of X_{25}	-	-	2^{58}
6	MK_4	SK_{98}	$(5,4)$ of X_{24}	$(0,?)$	8	2^{50}
7	-	WK_5, SK_{101}	$(3,2)$ of X_{25}	-	-	2^{50}
8	-	SK_{97}	$(3,2)$ of X_{24}	-	-	2^{50}
9	MK_8	SK_{93}	$(3,2)$ of X_{23}	$(0,?)$	8	2^{42}
10	-	SK_1	$(4,3)$ of X_1	-	-	2^{42}
11	MK_6	SK_6	$(6,5)$ of X_2	-	-	2^{42}
12	MK_{11}	SK_{11}	$(7,0)$ of X_3	$(?,0)$	8	2^{34}
13	-	WK_4, SK_{100}	$(1,0)$ of X_{25}	-	-	2^{34}
14	-	SK_{96}	$(1,0)$ of X_{24}	-	-	2^{34}
15	MK_{15}	SK_{92}	$(1,0)$ of X_{23}	-	-	2^{34}
16	-	SK_{88}	$(1,0)$ of X_{22}	$(0,e_{0,\sim})$	8	2^{26}
17	-	SK_0	$(2,1)$ of X_1	-	-	2^{26}
18	MK_5	SK_5	$(4,3)$ of X_2	-	-	2^{26}
19	MK_{10}	SK_{10}	$(6,5)$ of X_3	-	-	2^{26}
20	-	SK_{15}	$(7,0)$ of X_4	$(?,0)$	8	2^{18}
21	-	SK_{99}	$(7,6)$ of X_{24}	-	-	2^{18}
22	-	SK_{95}	$(7,6)$ of X_{23}	-	-	2^{18}
23	MK_{14}	SK_{91}	$(7,6)$ of X_{22}	-	-	2^{18}
24	-	SK_{87}	$(7,6)$ of X_{21}	$(0,e_7)$	7	2^{11}
25	-	SK_4	$(2,1)$ of X_2	-	-	2^{11}
26	MK_9	SK_9	$(4,3)$ of X_3	-	-	2^{11}
27	-	SK_{14}	$(6,5)$ of X_4	-	-	2^{11}
28	-	SK_{19}	$(7,0)$ of X_5	$(e_{0,\sim},0)$	8	-

3.5 Complexity of the Attack

In this section we give the data, memory and time complexities of our 26-round impossible differential attack on HIGHT.

3.5.1 Data Complexity

In order to apply the attack on HIGHT-26, we require 2^{13} structures of 2^{48} plaintexts. We divided these 2^{48} plaintexts into two sets having 2^{47} elements and if we take one element from both of the sets, we want their difference to be $(?, ?, ?, ?, ?, e_{0,\sim}, 0, 0)$. Hence the data complexity of this attack is $2^{13} \cdot 2^{48} = 2^{61}$ chosen-plaintexts.

3.5.2 Memory Complexity

At 28th step of the attack, if a pair satisfies the guessed key, it is eliminated. At the end of the attack, we have to know which keys are eliminated and which are not. Since the attack is on 112 bits of the secret key, namely $(MK_{15}, MK_{14}, MK_{11}, \ldots, MK_0)$, at the beginning of the attack we form an array of bits with size 2^{112} and initialize them to 0. The r-th element of the array represents the key whose integer value is r. When a key is eliminated, its corresponding value is converted to 1. Such an array requires 2^{112} bits which is 2^{109} bytes. During the attack, some values must be stored in the memory too but their size is negligible when compared to this array. Hence the memory complexity of the attack is 2^{109} bytes.

3.5.3 Time Complexity

We are going to use the abbreviation HE for reduced round HIGHT encryptions. In the first step, we guess MK_3 which can take 2^8 different values. We have 2^{82} pairs and we partially encrypt all of them with the guessed key. Hence we do $2 \cdot 2^8 \cdot 2^{82}$ partial encryptions in the first step (multiplication by 2 comes from the fact that we are working with pairs). However, we should compare the time complexity of the attack with exhaustive search which is 2^{128} HIGHT-26 encryptions. In this step we encrypted only 1/4th of a round of HIGHT. Hence the time complexity of the first step is $2 \cdot 2^8 \cdot 2^{82} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{84.30}$ HIGHT-26 encryptions.

In the second step , we guess MK_1 so the number of guessed key bits becomes 16. In the first step 2^8 of the pairs are eliminated and now 2^{74} pairs remain. Hence the time complexity of the second step is $2 \cdot 2^{16} \cdot 2^{74} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{84.30}$ HIGHT-26 encryptions. The time complexities of the first 27 steps are calculated in a similar way and they are given in Table 3.4.

In step 28, a pair eliminates 2^8 of the keys and once a key is eliminated, there is no need to use that key with other pairs. Hence for the first pair, we check 2^{112} keys, for the second pair we check $2^{112} - 2^{104} = 2^{112} \cdot (1 - 2^{-8})$ keys and so on and so forth. Thus the complexity of this step is $2 \cdot 2^{112} \left\{1 + (1 - 2^{-8}) + \ldots + (1 - 2^{-8})^{2^{11}-1}\right\} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{114.30}$ HIGHT-26 encryptions.

In step 29, we have approximately $2^{100.46}$ key candidates which covers the 112 bits of the secret key. For all these keys, we guess the remaining 16 bits of the key and check whether the obtained 128 bit key is correct or not. Hence the complexity of this step is $2^{100.46+16} = 2^{116.46}$ HIGHT-26 encryptions.

Time complexity of each step is given in Table 3.4. The total time complexity of the attack is the sum of the time complexities of each step, which is $2^{119.53}$ HIGHT-26 encryptions.

3.6 Summary

In this chapter we introduced our impossible differential attack on HIGHT-26 which uses a 16-round impossible differential characteristic. To our knowledge, this is the best attack on HIGHT among the attacks that does not use related-keys.

Impossible differential cryptanalysis results are given in Table 3.5. Our attack has lower time complexity than Lu's attack and requires less memory since we guessed 112 bits of the key instead of 120 bits during the attack.

Table 3.4: Time Complexities of Each Step of the Attack

Step	Complexity (HE)
1	$2 \cdot 2^{8} \cdot 2^{82} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{84.30}$
2	$2 \cdot 2^{16} \cdot 2^{74} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{84.30}$
3	$2 \cdot 2^{24} \cdot 2^{66} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{84.30}$
4	$2 \cdot 2^{32} \cdot 2^{66} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{92.30}$
5	$2 \cdot 2^{40} \cdot 2^{58} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{92.30}$
6	$2 \cdot 2^{48} \cdot 2^{58} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{100.30}$
7	$2 \cdot 2^{48} \cdot 2^{50} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{92.30}$
8	$2 \cdot 2^{48} \cdot 2^{50} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{92.30}$
9	$2 \cdot 2^{56} \cdot 2^{50} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{100.30}$
10	$2 \cdot 2^{56} \cdot 2^{50} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{92.30}$
11	$2 \cdot 2^{64} \cdot 2^{42} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{100.30}$
12	$2 \cdot 2^{72} \cdot 2^{42} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{108.30}$
13	$2 \cdot 2^{72} \cdot 2^{34} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{100.30}$
14	$2 \cdot 2^{72} \cdot 2^{34} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{100.30}$
15	$2 \cdot 2^{80} \cdot 2^{34} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{108.30}$
16	$2 \cdot 2^{80} \cdot 2^{34} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{108.30}$
17	$2 \cdot 2^{80} \cdot 2^{26} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{100.30}$
18	$2 \cdot 2^{88} \cdot 2^{26} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{108.30}$
19	$2 \cdot 2^{96} \cdot 2^{26} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{116.30}$
20	$2 \cdot 2^{96} \cdot 2^{26} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{116.30}$
21	$2 \cdot 2^{96} \cdot 2^{18} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{108.30}$
22	$2 \cdot 2^{96} \cdot 2^{18} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{108.30}$
23	$2 \cdot 2^{104} \cdot 2^{18} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{116.30}$
24	$2 \cdot 2^{104} \cdot 2^{18} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{116.30}$
25	$2 \cdot 2^{104} \cdot 2^{11} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{109.30}$
26	$2 \cdot 2^{112} \cdot 2^{11} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{117.30}$
27	$2 \cdot 2^{112} \cdot 2^{11} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{117.30}$
28	$2 \cdot 2^{112} \left\{1 + (1-2^{-8}) + \ldots + (1-2^{-8})^{2^{11}-1}\right\} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{114.30}$
29	$2^{100.46+16} = 2^{116.46}$
Total	$\approx 2^{119.53}$

Table 3.5: Summary of the impossible differential attacks on HIGHT

Rounds	Key Size	Data Complexity	Time Complexity	Memory Complexity	Reference
18	128	$2^{46.8}$ CP	$2^{109.2}$ HE	not specified	[2]
25	128	2^{60} CP	$2^{126.78}$ HE	not specified	[12]
26	128	2^{61} CP	$2^{119.53}$ HE	2^{109} bytes	Chapter 3

CHAPTER 4

RELATED-KEY IMPOSSIBLE DIFFERENTIAL CRYPTANALYSIS

In this chapter we introduce our related-key impossible differential attack on HIGHT-31 that uses a 22-round impossible differential characteristic and covers the rounds 0-30. The input whitening is excluded in our attack as done in [13]. To our knowledge, this is the best attack on HIGHT.

4.1 22-round Related-key Impossible Differential Characteristic

In Chapter 3 we showed that because of the Feistel-like structure of HIGHT, theoretically the highest round possible impossible differential characteristic is 18 rounds. However, we can increase this number by using related-keys. Idea is to give a difference to a secret key which would result in differences at certain subkeys and give the same difference to pairs so that they cancel out. This cancellation results in a few rounds in which pairs have no difference at all, until another difference comes from another affected subkey.

As described at Chapter 2, subkeys are obtained by modular addition of secret key bytes and some constants. Because of the modular addition, we will assume that a secret key byte has the difference e_7 so that the corresponding subkeys have also the same difference. Otherwise some bits of the differences of the subkeys would be undetermined which would prevent us to obtain differential characteristics that hold with probability 1. A difference in a secret key byte effects 8 subkeys and at most 1 whitening key which are given in Table 2.3.

We gave the difference e_7 to every 16 secret key byte and tried to obtain the highest round

Table 4.1: 22-Round Related-key Impossible Differential Characteristic, $\Delta MK_9 = e_7$

	$\Delta X_{i,7}$	$\Delta X_{i,6}$	$\Delta X_{i,5}$	$\Delta X_{i,4}$	$\Delta X_{i,3}$	$\Delta X_{i,2}$	$\Delta X_{i,1}$	$\Delta X_{i,0}$	Subkeys			
ΔX_6	0	0	e_7	0	0	0	0	0	SK_{27}	SK_{26}	SK_{25}	SK_{24}
ΔX_7	0	0	0	0	0	0	0	0	SK_{31}	SK_{30}	SK_{29}	SK_{28}
ΔX_8	0	0	0	0	0	0	0	0	SK_{35}	SK_{34}	SK_{33}	SK_{32}
ΔX_9	0	0	0	0	0	0	0	0	SK_{39}	SK_{38}	SK_{37}	SK_{36}
ΔX_{10}	0	0	0	0	0	0	0	0	SK_{43}	SK_{42}	SK_{41}	SK_{40}
ΔX_{11}	0	0	0	0	0	0	0	e_7	SK_{47}	SK_{46}	SK_{45}	SK_{44}
ΔX_{12}	0	0	0	0	0	$e_{2,\sim}$	e_7	0	SK_{51}	SK_{50}	SK_{49}	SK_{48}
ΔX_{13}	0	0	0	?	$e_{2,\sim}$	e_7	0	0	SK_{55}	SK_{54}	SK_{53}	SK_{52}
ΔX_{14}	0	?	?	$e_{0,\sim}$	e_7	0	0	0	SK_{59}	SK_{58}	SK_{57}	SK_{56}
ΔX_{15}	?	?	$e_{0,\sim}$	e_7	0	0	0	?	SK_{63}	SK_{62}	SK_{61}	SK_{60}
ΔX_{16}	?	$e_{0,\sim}$	e_7	0	0	?	?	?	SK_{67}	SK_{66}	SK_{65}	SK_{64}
ΔX_{17}	$e_{0,\sim}$	e_7	0	?	?	?	?	?	SK_{71}	SK_{70}	SK_{69}	SK_{68}
ΔX_{18}	e_7	?	?	?	?	?	?	$e_{\tilde{0},\sim}$	SK_{75}	SK_{74}	SK_{73}	SK_{72}
ΔX_{18}	0	0	?	?	?	?	$e_{\tilde{0},\sim}$	e_7	SK_{75}	SK_{74}	SK_{73}	SK_{72}
ΔX_{19}	0	0	?	?	?	$e_{0,\sim}$	e_7	0	SK_{79}	SK_{78}	SK_{77}	SK_{76}
ΔX_{20}	0	0	?	?	$e_{0,\sim}$	e_7	0	0	SK_{83}	SK_{82}	SK_{81}	SK_{80}
ΔX_{21}	0	0	?	$e_{2,\sim}$	e_7	0	0	0	SK_{87}	SK_{86}	SK_{85}	SK_{84}
ΔX_{22}	0	0	$e_{2,\sim}$	e_7	0	0	0	0	SK_{91}	SK_{90}	SK_{89}	SK_{88}
ΔX_{23}	0	0	e_7	0	0	0	0	0	SK_{95}	SK_{94}	SK_{93}	SK_{92}
ΔX_{24}	0	0	0	0	0	0	0	0	SK_{99}	SK_{98}	SK_{97}	SK_{96}
ΔX_{25}	0	0	0	0	0	0	0	0	SK_{103}	SK_{102}	SK_{101}	SK_{100}
ΔX_{26}	0	0	0	0	0	0	0	0	SK_{107}	SK_{106}	SK_{105}	SK_{104}
ΔX_{27}	0	0	0	0	0	0	0	0	SK_{111}	SK_{110}	SK_{109}	SK_{108}
ΔX_{28}	0	0	0	0	0	0	0	e_7	SK_{115}	SK_{114}	SK_{113}	SK_{112}

related-key impossible differential characteristic by giving the difference e_7 to pairs from above and belove and cancel them with subkey difference. We observed that the highest round characteristic that can be obtained with this method is 22 rounds and it can be done by giving the difference e_7 to either MK_{15} or MK_9.

In our 31-round related key impossible differential attack, we gave difference to MK_{15} and it will be shown in detail in the next section. The 22-round related key impossible differential characteristic which is obtained by giving the difference e_7 to MK_9, which affects SK_9, SK_{26}, SK_{43}, SK_{60}, SK_{77}, SK_{94}, SK_{111} and SK_{120}, is shown in Table 4.1 where the bytes which cause the differentials to miss in the middle are denoted with a light gray background and the subkeys which are affected by the key difference are denoted with a dark gray background.

4.2 31-round Path

We obtained a 22-round related-key impossible differential characteristic by giving the difference e_7 to MK_{15}, which also gives the same difference to WK_3, SK_{15}, SK_{24}, SK_{41}, SK_{58}, SK_{75}, SK_{92}, SK_{109} and SK_{126}. We extend this characteristic to a 31-round related-key impossible differential path by adding 6 rounds to above of the characteristic and 3 rounds to

below of the characteristic. This path is shown in Table 4.2 where the bytes which cause the differentials to miss in the middle are denoted with a light gray background and the subkeys which are affected by the key difference are denoted with a dark gray background.

A similar 31-round related-key impossible differential path can be obtained by extending the 22-round related-key impossible differential characteristic that is constructed by the key difference e_7 in MK_9. The steps and the complexity of such an attack is similar to the one that we are going to define in the following sections.

4.3 Data Collection

1. We choose 2^{15} structures of 2^{49} plaintexts P^i each where the byte (2) and the least significant seven bits of the byte (3) are fixed to certain values. The bytes $(7, 6, 5, 1, 0)$ and the most significant seven bits of the byte (4) contain every possible values (2^{47} of these plaintexts have 1 in their first and the seventh bit of their bytes $(4, 3)$ respectively and 2^{47} of them have 0 at these places).

    - One structure of chosen plaintexts proposes exactly 2^{94} pairs and we obtain 2^{109} pairs in total. If we change the structure type as 2^{47} plaintexts having 0 as the least significant bit of byte (4) and 1 as the most significant bit of byte (3) and similarly 2^{47} plaintexts having 1 as the least significant bit of byte (4) and 0 as the most significant bit of byte (3), these structures also proposes 2^{94} pairs. We choose 2^{15} such structures and obtain 2^{110} plaintext pairs in total.

2. We obtain all the ciphertexts C^i of the plaintexts P^i encrypted with K^1 and ciphertext $C^{i'}$ of the plaintexts P^i encrypted with K^2 where $K^1 \oplus K^2 = (e_7, 0, \ldots, 0)$. We choose only the ciphertext pairs $(C^i, C^{j'})$ satisfying the difference $(e_{0,\sim}, e_7, 0, 0, 0, 0, ?, ?)$.

    - This step can be done by inserting all the ciphertexts into a hash table indexed by expected inactive bits and choosing the colliding pairs which satisfy the required difference. There is 41-bit filtering condition over the ciphertext pairs. Therefore 2^{69} pairs remain.

3. This attack will be on 120 bits of the secret key, namely (MK_{15}, MK_{14}, MK_{12}, ... , MK_0) and since we are going to eliminate wrong keys, we have to store them. One trivial way of doing this is by forming an array of bits with size 2^{120} and initialize them

Table 4.2: 31-Round Related-Key Impossible Differential

	$\Delta X_{i,7}$	$\Delta X_{i,6}$	$\Delta X_{i,5}$	$\Delta X_{i,4}$	$\Delta X_{i,3}$	$\Delta X_{i,2}$	$\Delta X_{i,1}$	$\Delta X_{i,0}$	Subkeys			
ΔX_0	?	?	?	$e_{0,\sim}$	e_7	0	?	?	SK_3	SK_2	SK_1	SK_0
ΔX_1	?	?	$e_{0,\sim}$	e_7	0	0	?	?	SK_7	SK_6	SK_5	SK_4
ΔX_2	?	$e_{0,\sim}$	e_7	0	0	0	?	?	SK_{11}	SK_{10}	SK_9	SK_8
ΔX_3	$e_{0,\sim}$	e_7	0	0	0	0	?	?	SK_{15}	SK_{14}	SK_{13}	SK_{12}
ΔX_4	e_7	0	0	0	0	0	?	$e_{2,\sim}$	SK_{19}	SK_{18}	SK_{17}	SK_{16}
ΔX_5	0	0	0	0	0	0	$e_{2,\sim}$	e_7	SK_{23}	SK_{22}	SK_{21}	SK_{20}
ΔX_6	0	0	0	0	0	0	e_7	0	SK_{27}	SK_{26}	SK_{25}	SK_{24}
ΔX_7	0	0	0	0	0	0	0	0	SK_{31}	SK_{30}	SK_{29}	SK_{28}
ΔX_8	0	0	0	0	0	0	0	0	SK_{35}	SK_{34}	SK_{33}	SK_{32}
ΔX_9	0	0	0	0	0	0	0	0	SK_{39}	SK_{38}	SK_{37}	SK_{36}
ΔX_{10}	0	0	0	0	0	0	0	0	SK_{43}	SK_{42}	SK_{41}	SK_{40}
ΔX_{11}	0	0	0	e_7	0	0	0	0	SK_{47}	SK_{46}	SK_{45}	SK_{44}
ΔX_{12}	0	$e_{2,\sim}$	e_7	0	0	0	0	0	SK_{51}	SK_{50}	SK_{49}	SK_{48}
ΔX_{13}	$e_{2,\sim}$	e_7	0	0	0	0	0	?	SK_{55}	SK_{54}	SK_{53}	SK_{52}
ΔX_{14}	e_7	0	0	0	0	?	?	$e_{0,\sim}$	SK_{59}	SK_{58}	SK_{57}	SK_{56}
ΔX_{15}	0	e_7	0	?	?	?	$e_{0,\sim}$	e_7	SK_{63}	SK_{62}	SK_{61}	SK_{60}
ΔX_{16}	e_7	?	?	?	?	$e_{0,\sim}$	e_7	$e_{0,\sim}$	SK_{67}	SK_{66}	SK_{65}	SK_{64}
ΔX_{17}	?	?	?	?	$e_{0,\sim}$	?	$e_{0,\sim}$	?	SK_{71}	SK_{70}	SK_{69}	SK_{68}
ΔX_{17}	?	?	?	$e_{0,\sim}$	e_7	0	?	?	SK_{71}	SK_{70}	SK_{69}	SK_{68}
ΔX_{18}	?	?	$e_{0,\sim}$	e_7	0	0	?	?	SK_{75}	SK_{74}	SK_{73}	SK_{72}
ΔX_{19}	?	$e_{0,\sim}$	e_7	0	0	0	?	?	SK_{79}	SK_{78}	SK_{77}	SK_{76}
ΔX_{20}	$e_{0,\sim}$	e_7	0	0	0	0	?	?	SK_{83}	SK_{82}	SK_{81}	SK_{80}
ΔX_{21}	e_7	0	0	0	0	0	?	$e_{2,\sim}$	SK_{87}	SK_{86}	SK_{85}	SK_{84}
ΔX_{22}	0	0	0	0	0	0	$e_{2,\sim}$	e_7	SK_{91}	SK_{90}	SK_{89}	SK_{88}
ΔX_{23}	0	0	0	0	0	0	e_7	0	SK_{95}	SK_{94}	SK_{93}	SK_{92}
ΔX_{24}	0	0	0	0	0	0	0	0	SK_{99}	SK_{98}	SK_{97}	SK_{96}
ΔX_{25}	0	0	0	0	0	0	0	0	SK_{103}	SK_{102}	SK_{101}	SK_{100}
ΔX_{26}	0	0	0	0	0	0	0	0	SK_{107}	SK_{106}	SK_{105}	SK_{104}
ΔX_{27}	0	0	0	0	0	0	0	0	SK_{111}	SK_{110}	SK_{109}	SK_{108}
ΔX_{28}	0	0	0	e_7	0	0	0	0	SK_{115}	SK_{114}	SK_{113}	SK_{112}
ΔX_{29}	0	$e_{2,\sim}$	e_7	0	0	0	0	0	SK_{119}	SK_{118}	SK_{117}	SK_{116}
ΔX_{30}	$e_{2,\sim}$	e_7	0	0	0	0	0	?	SK_{123}	SK_{122}	SK_{121}	SK_{120}
ΔX_{31}	e_7	0	0	0	0	?	?	$e_{0,\sim}$	WK_7	WK_6	WK_5	WK_4
FT	$e_{0,\sim}$	e_7	0	0	0	0	?	?				

to 0. The r-th element of the array represents the key whose integer value is r. When a key is eliminated, we will convert its corresponding value in the array to 1.

4.4 Related-Key Impossible Differential Attack on HIGHT-31

Attack procedure is to take a pair and perform partial encryptions and decryptions by guessing the necessary key bytes and eliminating them if they satisfy the impossible differential characteristic. The following steps are applied to every pair that we obtained in the data collection part.

1. We guess MK_0 and get SK_0 (actually by guessing MK_0, we get WK_4, SK_0, SK_{17}, SK_{34}, SK_{51}, SK_{68}, SK_{85}, SK_{102} and SK_{119}, which can be checked from Table 2.3). Since we know the values of the plaintexts, we obtain the bytes $(2,1)$ of X_1 by partially encrypting every plaintexts' bytes $(1,0)$ of X_0 with SK_0. In order to satisfy the impossible differential characteristic, the obtained values of the pairs must have the difference $(0,?)$. We keep the pairs satisfying this condition. Since the difference $(0,?)$ contains 8-bit filtering condition, 2^8 of the pairs are eliminated in this step. Hence $2^{69}/2^8 = 2^{61}$ pairs remain.

2. We guess MK_3 and obtain the bytes $(7,0)$ of X_1 by partially encrypting the bytes $(7,6)$ of X_0 with SK_3 for all remaining pairs. There are no filtering conditions in this step since the required difference is $(?,?)$.

3. We guess MK_4 and obtain the bytes $(2,1)$ of X_2 by partially encrypting the bytes $(1,0)$ of X_1 with SK_4 for all remaining pairs. We keep the pairs having the difference $(0,?)$ at the bytes $(2,1)$ of X_2. It contains 8-bit filtering condition so 2^{53} pairs remain.

4. We guess MK_9 and obtain the bytes $(1,0)$ of X_{30} by partially decrypting the bytes $(2,1)$ of X_{31} with WK_4 and SK_{120} for all remaining pairs. We keep the pairs having the difference $(0,?)$ at the bytes $(1,0)$ of X_{30}. It contains 8-bit filtering condition so 2^{45} pairs remain.

5. We guess MK_{12} and obtain the bytes $(7,6)$ of X_{30} by partially decrypting the bytes $(7,0)$ of X_{31} with WK_7 and SK_{123} for all remaining pairs. We keep the pairs having the difference $(e_{2,\sim}, e_7)$ at the bytes $(7,6)$ of X_{30}. It contains 2-bit filtering condition

because $F_0(e_7)$ becomes $e_{0,\sim}$ after modular addition and byte (0) of X_{31} is $e_{0,\sim}$. So the least significant bit of byte (7) of X_{30} is always 0. Thus the filtering conditions are on bits (2, 1) of byte (7) of X_{30}. Hence 2^{43} pairs remain.

6. We obtain the bytes $(7, 6)$ of X_{29} by partially decrypting the bytes $(7, 0)$ of X_{30} with SK_{119} for all remaining pairs. Since we already guessed the secret key byte MK_0, we know the value of SK_{119}. We keep the pairs having the difference $(0, e_{2,\sim})$ at the bytes $(7, 6)$ of X_{29}. It contains 8-bit filtering condition so 2^{35} pairs remain.

7. We guess MK_2 and obtain the bytes $(6, 5)$ of X_1 by partially encrypting the bytes $(5, 4)$ of X_0 with SK_2 for all remaining pairs. There are no filtering conditions in this step.

8. We guess MK_7 and obtain the bytes $(7, 0)$ of X_2 by partially encrypting the bytes $(7, 6)$ of X_1 with SK_7 for all remaining pairs. There are no filtering conditions in this step.

9. We guess MK_8 and obtain the bytes $(2, 1)$ of X_3 by partially encrypting the bytes $(1, 0)$ of X_2 with SK_8 for all remaining pairs. We keep the pairs having the difference $(0, ?)$ at the bytes $(2, 1)$ of X_3. It contains 8-bit filtering condition so 2^{27} pairs remain.

10. We guess MK_{11} and obtain the bytes $(5, 4)$ of X_{30} by partially decrypting the bytes $(6, 5)$ of X_{31} with WK_6 and SK_{122} for all remaining pairs. Since we already guessed the secret key byte MK_2, we know the value of WK_6. There are no filtering conditions in this step.

11. We obtain the bytes $(5, 4)$ of X_{29} by partially decrypting the bytes $(6, 5)$ of X_{30} with SK_{118} for all remaining pairs. Since we already guessed the secret key byte MK_7, we know the value of SK_{118}.There are no filtering conditions in this step.

12. We obtain the bytes $(5, 4)$ of X_{28} by partially decrypting the bytes $(6, 5)$ of X_{29} with SK_{114} for all remaining pairs. We keep the pairs having the difference $(0, e_7)$ at the bytes $(5, 4)$ of X_{28}. It contains 5-bit filtering condition because $F_1(e_7)$ becomes $e_{2,\sim}$ after modular addition and byte (5) of X_{28} becomes $e_{2,\sim} \oplus e_{2,\sim} = e_{\bar{2},\sim}$. So the least significant 3 bits are always 0. Thus the filtering conditions are on the remaining 5 bits of byte (5) of X_{28}. Hence 2^{22} pairs remain.

13. We guess MK_1 and obtain the bytes $(4, 3)$ of X_1 by partially encrypting the bytes $(3, 2)$ of X_0 with SK_1 for all remaining pairs. There are no filtering conditions in this step.

14. We guess MK_6 and obtain the bytes $(6,5)$ of X_2 by partially encrypting the bytes $(5,4)$ of X_1 with SK_6 for all remaining pairs. There are no filtering conditions in this step.

15. We obtain the bytes $(7,0)$ of X_3 by partially encrypting $(7,6)$ of X_2 with SK_{11} for all remaining pairs. Since we already guessed the secret key byte MK_{11}, we know the value of SK_{11}. There are no filtering conditions in this step.

16. We obtain the bytes $(2,1)$ of X_4 by partially encrypting $(1,0)$ of X_3 with SK_{12} for all remaining pairs. Since we already guessed the secret key byte MK_{12}, we know the value of SK_{12}. We keep the pairs having the difference $(0,?)$ at the bytes $(2,1)$ of X_4. It contains 8-bit filtering condition so 2^{14} pairs remain.

17. We guess MK_5 and obtain the bytes $(4,3)$ of X_2 by partially encrypting the bytes $(3,2)$ of X_1 with SK_5 for all remaining pairs. There are no filtering conditions in this step.

18. We guess MK_{10} and obtain the bytes $(6,5)$ of X_3 by partially encrypting the bytes $(5,4)$ of X_2 with SK_{10} for all remaining pairs. There are no filtering conditions in this step.

19. We guess MK_{15} and obtain the bytes $(7,0)$ of X_4 by partially encrypting the bytes $(7,6)$ of X_3 with SK_{15} for all remaining pairs. We keep the pairs having the difference $(e_7, e_{2,\sim})$ at the bytes $(7,0)$ of X_4. It contains 2-bit filtering condition because $F_0(e_7)$ becomes $e_{0,\sim}$ after modular addition and byte (7) of X_3 is $e_{0,\sim}$. So the least significant bit of byte (0) of X_4 is always 0. Thus the filtering conditions are on bits $(2,1)$ of byte (0) of X_4. Hence 2^{12} pairs remain.

20. We obtain the bytes $(2,1)$ of X_5 by partially encrypting $(1,0)$ of X_4 with SK_{16} for all remaining pairs. Since we already guessed the secret key byte MK_7, we know the value of SK_{16}. We keep the pairs having the difference $(0, e_{0,\sim})$ at the bytes $(2,1)$ of X_5. It contains 8-bit filtering condition so 2^4 pairs remain.

21. We obtain the bytes $(4,3)$ of X_3 by partially encrypting $(3,2)$ of X_2 with SK_9 for all remaining pairs. Since we already guessed the secret key byte MK_9, we know the value of SK_9. There are no filtering conditions in this step.

22. We guess MK_{14} and obtain the bytes $(6,5)$ of X_4 by partially encrypting the bytes $(5,4)$ of X_3 with SK_{14} for all remaining pairs. There are no filtering conditions in this step.

23. We obtain the bytes $(7, 0)$ of X_5 by partially encrypting $(7, 6)$ of X_4 with SK_{19} for all remaining pairs. Since we already guessed the secret key byte MK_2, we know the value of SK_{19}. There are no filtering conditions in this step.

24. We obtain the bytes $(2, 1)$ of X_6 by partially encrypting $(1, 0)$ of X_5 with SK_{20} for all remaining pairs. Since we already guessed the secret key byte MK_3, we know the value of SK_{20}. We check if the obtained difference is $(0, e_7)$ at the bytes $(2, 1)$ of X_6. If a pair satisfies this condition, we eliminate the corresponding key. It contains 5-bit filtering condition because $F_1(e_7)$ becomes $e_{2,\sim}$ after modular addition and byte (2) of X_6 becomes $e_{2,\sim} \oplus e_{2,\sim} = e_{\bar{2},\sim}$. So the least significant 3 bits are always 0. Thus the filtering conditions are on the remaining 5 bits of byte (2) of X_6. Since there is an 5-bit condition, every pair eliminates 2^{-5} of the keys. Therefore after the first pair, there remain $2^{120} - 2^{115} = 2^{120} \cdot (1 - 2^{-5})$ keys. After the second pair, it is expected to have $2^{120} \cdot (1 - 2^{-5}) - 2^{120} \cdot (1 - 2^{-5}) \cdot 2^{-5} = 2^{120} \cdot (1 - 2^{-5})^2$ many keys. Following that manner, after the last pair, we have $2^{120} \cdot (1 - 2^{-5})^{2^4} \approx 2^{119.27}$ remaining keys.

25. Now we reduced the number of possible keys from 2^{120} to $2^{119.27}$. For all these remaining keys, we guess the remaining 8 bits of the key, that is MK_{13}, and exhaustively search for the correct one. Hence we check $2^{119.27+8} = 2^{127.27}$ keys in this step. This is done by encrypting one plaintext with the guessed key and checking whether the corresponding ciphertext is obtained. To avoid false alarms, if a correct ciphertext is obtained with a key, an another plaintext-ciphertext pair should be checked with the same key. If the correct ciphertext is obtained again, we do the same with a third pair. If the correct ciphertext is obtained for the third time, we conclude that the guessed key is the correct secret key because the probability of obtaining the correct ciphertexts three times with a wrong key is $(2^{-64})^3 = 2^{-192}$ and expected number of wrong keys is $2^{-192} \cdot 2^{127.27} = 2^{-64.73}$.

These steps are summarized in Table 4.3.

4.5 Complexity

In this section we give the data, memory and time complexities of our related-key impossible differential attack on HIGHT-31.

Table 4.3: 31-Round related key impossible differential attack, $\Delta MK_{15} = e_7$

	Guess Key Byte	Use	Obtain	Check Difference	Condition (In terms of bits)	Remaining Pairs
1	MK_0	SK_0	$(2,1)$ of X_1	$(0,?)$	8 bits	2^{61}
2	MK_3	SK_3	$(7,0)$ of X_1	-	-	2^{61}
3	MK_4	SK_4	$(2,1)$ of X_2	$(0,?)$	8 bits	2^{53}
4	MK_9	WK_4, SK_{120}	$(1,0)$ of X_{30}	$(0,?)$	8 bits	2^{45}
5	MK_{12}	WK_7, SK_{123}	$(7,6)$ of X_{30}	$(e_{2,\sim}, e_7)$	2 bits	2^{43}
6	-	SK_{119}	$(7,6)$ of X_{29}	$(0, e_{2,\sim})$	8 bits	2^{35}
7	MK_2	SK_2	$(6,5)$ of X_1	-	-	2^{35}
8	MK_7	SK_7	$(7,0)$ of X_2	-	-	2^{35}
9	MK_8	SK_8	$(2,1)$ of X_3	$(0,?)$	8 bits	2^{27}
10	MK_{11}	WK_6, SK_{122}	$(5,4)$ of X_{30}	-	-	2^{27}
11	-	SK_{118}	$(5,4)$ of X_{29}	-	-	2^{27}
12	-	SK_{114}	$(5,4)$ of X_{28}	$(0, e_7)$	5 bits	2^{22}
13	MK_1	SK_1	$(4,3)$ of X_1	-	-	2^{22}
14	MK_6	SK_6	$(6,5)$ of X_2	-	-	2^{22}
15	-	SK_{11}	$(7,0)$ of X_3	-	-	2^{22}
16	-	SK_{12}	$(2,1)$ of X_4	$(0,?)$	8 bits	2^{14}
17	MK_5	SK_5	$(4,3)$ of X_2	-	-	2^{14}
18	MK_{10}	SK_{10}	$(6,5)$ of X_3	-	-	2^{14}
19	MK_{15}	SK_{15}	$(7,0)$ of X_4	$(e_7, e_{2,\sim})$	2 bits	2^{12}
20	-	SK_{16}	$(2,1)$ of X_5	$(0, e_{2,\sim})$	8 bits	2^4
21	-	SK_9	$(4,3)$ of X_3	-	-	2^4
22	MK_{14}	SK_{14}	$(6,5)$ of X_4	-	-	2^4
23	-	SK_{19}	$(7,0)$ of X_5	-	-	2^4
24	-	SK_{20}	$(2,1)$ of X_6	$(0, e_7)$	5 bits	-

4.5.1 Data Complexity

Since the block size of HIGHT is 64, we can have at most 2^{64} different plaintext blocks and we used all of them in our attack. However, 2^{63} of these plaintexts are encrypted with K^1 and the other with K^2. This attack has more filtering conditions than our attack on HIGHT-26 that is given in Chapter 3 and this is the reason for taking more plaintexts in the data collection part. The data complexity of this attack is 2^{64} chosen-plaintexts. However, the number of pairs can still be increased in order to decrease the time complexity or the attack can be applied with less data complexity which results in higher time complexity. These situations are considered in Section 4.6.

4.5.2 Memory Complexity

At 24th step of the attack, if a pair satisfies the guessed key, it is eliminated. At the end of the attack, we have to know which keys are eliminated and which are not. Since the attack is on 120 bits of the secret key, namely $(MK_{15}, MK_{14}, MK_{12}, \ldots, MK_0)$, at the beginning of the

attack we form an array of bits with size 2^{120} and initialize them to 0. The r-th element of the array represents the key whose integer value is r. When a key is eliminated, its corresponding value is converted to 1.

Such an array requires 2^{120} bits which is 2^{117} bytes. During the attack, some values must be stored in the memory too but their size is negligible when compared to this array. Hence the memory complexity of the attack is 2^{117} bytes.

4.5.3 Time Complexity

In the first step, we guess MK_0 which can take 2^8 different values. We have 2^{69} pairs and we partially encrypt all of them with the guessed key. Hence we do $2 \cdot 2^8 \cdot 2^{69}$ partial encryptions in the first step (multiplication by 2 comes from the fact that we are working with pairs). However, we will compare the time complexity of the attack with exhaustive search which is 2^{128} HIGHT-31 encryptions. In this step we encrypted only 1/4th of a round of HIGHT. Hence the time complexity of the first step is $2 \cdot 2^8 \cdot 2^{69} \cdot \frac{1}{4} \cdot \frac{1}{31} \approx 2^{71.05}$ HIGHT-31 encryptions.

In the second step , we guess MK_3 so the number of guessed key bits becomes 16. In the first step 2^8 of the pairs are eliminated and now 2^{61} pairs remain. Hence the time complexity of the second step is $2 \cdot 2^{16} \cdot 2^{61} \cdot \frac{1}{4} \cdot \frac{1}{31} \approx 2^{71.05}$ HIGHT-31 encryptions. The time complexities of the first 23 steps are calculated in a similar way and they are given in Table 4.4.

In step 24, a pair eliminates 2^5 of the keys and once a key is eliminated, there is no need to use that key with other pairs. Hence for the first pair, we check 2^{120} keys, $2^{120} - 2^{115} = 2^{120} \cdot (1 - 2^{-5})$ keys are checked for the second pair and so on and so forth. Thus the complexity of this step is $2 \cdot 2^{120} \left\{1 + (1 - 2^{-5}) + \ldots + (1 - 2^{-5})^{2^4 - 1}\right\} \cdot \frac{1}{4} \cdot \frac{1}{31} \approx 2^{117.89}$ HIGHT-31 encryptions.

In step 25, we have approximately $2^{119.27}$ key candidates which covers the 120 bits of the secret key. For all these keys, we guess the remaining 8 bits of the key and check whether the obtained 128 bit key is correct or not. Hence the complexity of this step is $2^{119.17+8} = 2^{127.17}$ HIGHT-31 encryptions.

Time complexity of each step is given in Table 4.4. The total time complexity of the attack is the sum of the time complexities of each step, which is $2^{127.28}$ HIGHT-31 encryptions. However, this number can be decreased if we increase the number of pairs that we used in our

Table 4.4: Time Complexities of Each Step of the Attack

Step	Complexity (HE)
1	$2 \cdot 2^{8} \cdot 2^{69} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{84.30}$
2	$2 \cdot 2^{16} \cdot 2^{61} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{84.30}$
3	$2 \cdot 2^{24} \cdot 2^{61} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{84.30}$
4	$2 \cdot 2^{32} \cdot 2^{53} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{92.30}$
5	$2 \cdot 2^{40} \cdot 2^{45} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{92.30}$
6	$2 \cdot 2^{40} \cdot 2^{43} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{100.30}$
7	$2 \cdot 2^{48} \cdot 2^{35} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{92.30}$
8	$2 \cdot 2^{56} \cdot 2^{35} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{92.30}$
9	$2 \cdot 2^{64} \cdot 2^{35} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{100.30}$
10	$2 \cdot 2^{72} \cdot 2^{27} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{92.30}$
11	$2 \cdot 2^{72} \cdot 2^{27} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{100.30}$
12	$2 \cdot 2^{72} \cdot 2^{27} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{108.30}$
13	$2 \cdot 2^{80} \cdot 2^{22} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{100.30}$
14	$2 \cdot 2^{88} \cdot 2^{22} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{100.30}$
15	$2 \cdot 2^{88} \cdot 2^{22} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{108.30}$
16	$2 \cdot 2^{88} \cdot 2^{22} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{108.30}$
17	$2 \cdot 2^{96} \cdot 2^{14} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{100.30}$
18	$2 \cdot 2^{104} \cdot 2^{14} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{108.30}$
19	$2 \cdot 2^{112} \cdot 2^{14} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{116.30}$
20	$2 \cdot 2^{112} \cdot 2^{12} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{116.30}$
21	$2 \cdot 2^{112} \cdot 2^{4} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{108.30}$
22	$2 \cdot 2^{120} \cdot 2^{4} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{108.30}$
23	$2 \cdot 2^{120} \cdot 2^{4} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{116.30}$
24	$2 \cdot 2^{120} \left\{1 + (1-2^{-5}) + \ldots + (1-2^{-5})^{2^4-1}\right\} \cdot \frac{1}{4} \cdot \frac{1}{31} \approx 2^{114.30}$
25	$2^{119.27+8} = 2^{127.27}$
Total	$\approx 2^{127.28}$

attack and this situation is considered in Section 4.6.

4.6 Summary

In this chapter we introduced our related-key impossible differential attack on HIGHT-31 which uses a 22-round impossible differential characteristic. To our knowledge, this is the best attack on HIGHT.

Related-key impossible differential cryptanalysis results are given in Table 4.5. Our attack is slightly better than the exhaustive search and it reduces the security margin of HIGHT to 1.

Notice that in a related key attack, the pair (P^1, P^2) encrypted with the keys (K^1, K^2) and

Table 4.5: Summary of the related-key impossible differential attacks on HIGHT

Rounds	Key Size	Data Complexity	Time Complexity	Memory Complexity	Reference
28	128	2^{60} CP	$2^{125.54}$HE	not specified	[12]
31	128	2^{64} CP	$2^{127.28}$HE	2^{117} bytes	Chapter 4

the pair (P^2, P^1) encrypted with keys (K^1, K^2) are different since the corresponding ciphertexts are different. Hence the number of pairs can be doubled in this way. In that case, the time complexity of the attack reduces to $2^{126.56}$ HIGHT-31 encryptions and data complexity increases to 2^{65} CP.

Although we used 2^{64} chosen-plaintexts in our attack, the same attack can be applied with less data. For example, in the data collection part, if we choose 2^{14} structures instead of 2^{15}, the attack can be applied with 2^{63} data complexity, 2^{120} memory complexity and $2^{127.63}$ time complexity.

CHAPTER 5

CONCLUSION

In this study, we focused on the lightweight block cipher HIGHT and further investigated its security against impossible differential cryptanalysis. At first, we defined the (related-key) impossible differential cryptanalysis technique and presented the specifications of HIGHT and previous impossible differential attacks on HIGHT. Then we presented our impossible differential attack on HIGHT-26 (without initial transformation) which is the best attack on HIGHT among the attacks that does not use related-keys. This attack was an improvement to Lu's impossible differential attack on HIGHT-25 and although the both attacks use a 16-round impossible differential characteristic, our attack's time complexity is significantly better.

In Chapter 3 we also justify our steps when we are constructing our impossible differential characteristic and show why we believe that a better impossible differential characteristic can not be found and therefore an impossible differential attack can not be directly applicable to HIGHT-27 or higher.

In Chapter 4, we investigated the weaknesses in the key schedule of HIGHT and obtained two different 22-round related-key impossible differential characteristics. By using one of these characteristics, we presented our 31-round related-key impossible differential attack on HIGHT-31 (without initial transformation) which is slightly faster than exhaustive search. This is the best known cryptanalytic result on HIGHT.

The two attacks that are presented in this thesis are appeared in 14th Australasian Conference on Information Security and Privacy (ACISP 2009) with the title "Lightweight Block Ciphers Revisited: Cryptanalysis of Reduced Round PRESENT and HIGHT" [24] which is a collaborative work of Onur Özen, Kerem Varıcı, Cihangir Tezcan and Çelebi Kocair.

Table 5.1: Summary of the known attacks on HIGHT

Rounds	Attack Type	Data Complexity	Time Complexity	Memory Complexity	Reference
13	Differential	2^{62} CP	not specified	not specified	[2]
13	Linear	2^{57} KP	not specified	not specified	[2]
13	Boomerang	2^{62} CP	not specified	not specified	[2]
16	Truncated Differential	$2^{14.1}$ CP	$2^{108.69}$ HE	not specified	[2]
18	Impossible Differential	$2^{46.8}$ CP	$2^{109.2}$ HE	not specified	[2]
25	Impossible Differential	2^{60} CP	$2^{126.78}$HE	not specified	[12]
26	Impossible Differential	2^{61} CP	$2^{119.53}$HE	2^{109} bytes	Chapter 3
19	Related-key Boomerang	not specified	not specified	not specified	[2]
25	Related-key Rectangle	$2^{51.2}$ CP	$2^{120.41}$ HE	not specified	[12]
28	Related-key Impossible Diff.	2^{60} CP	$2^{125.54}$HE	not specified	[12]
31	Related-key Impossible Diff.	2^{64} CP	$2^{127.28}$HE	2^{117} bytes	Chapter 4

Summary of the known attacks on HIGHT is given in Table 5.1.

REFERENCES

[1] Joan Daemen and Vincent Rijmen. *The Design of Rijndael*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[2] Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bonseok Koo, Changhoon Lee, Donghoon Chang, Jaesang Lee, Kitae Jeong, Hyun Kim, Jongsung Kim, and Seongtaek Chee. *HIGHT: A New Block Cipher Suitable for Low-Resource Device*. In Louis Goubin and Mitsuru Matsui (editors), *CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 46-59. Springer, 2006.

[3] South Korea Telecommunications Technology Associations (TTA). *64-bit Block Cipher HIGHT*. Standardization Number TTAS.KO-12.0040, 27 December 2006, http://www.tta.or.kr/English/new/standardization/eng_ttastddesc.jsp?stdno=TTAK.KO-12.0040/R1, last visited on July 2009.

[4] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. *PRESENT: An Ultra-Lightweight Block Cipher*. In Pascal Paillier and Ingrid Verbauwhede (editors), *CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450-466. Springer, 2007.

[5] Chae Hoon Lim and Tymur Korkishko. *mCrypton - A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors*. In JooSeok Song, Taekyoung Kwon, and Moti Yung (editors), *WISA 2005*, volume 3786 of *Lecture Notes in Computer Science*, pages 243-258. Springer, 2005.

[6] François-Xavier Standaert, Gilles Piret, Neil Gershenfeld, and Jean-Jacques Quisquater. *SEA: A Scalable Encryption Algorithm for Small Embedded Applications*. In Josep Domingo-Ferrer, Joachim Posegga, and Daniel Schreckling (editors), *CARDIS 2006*, volume 3928 of *Lecture Notes in Computer Science*, pages 222-236. Springer, 2006.

[7] Matthew J. B. Robshaw. *Searching for Compact Algorithms: CGEN*. In Phong Q. Nguyen (editor), *VIETCRYPT*, volume 4341 of *Lecture Notes in Computer Science*, pages 37-49. Springer, 2006.

[8] Gregor Leander, Christof Paar, Axel Poschmann, and Kai Schramm. *New Lightweight DES Variants*. In Alex Biryukov (editor), *FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 196-210. Springer, 2007.

[9] David J. Wheeler and Roger M. Needham. *TEA, a Tiny Encryption Algorithm*. In Bart Preneel (editor), *FSE '94*, volume 1008 of *Lecture Notes in Computer Science*, pages 363-366. Springer, 1994.

[10] David J. Wheeler and Roger M. Needham. *TEA Extensions*. October 1997.

[11] Eli Biham. *New Types of Cryptanalytic Attacks Using Related Keys*. *Journal of Cryptology*, 7(4), pages 229-246, 1994.

[12] Jiqiang Lu. *Cryptanalysis of Block Ciphers*. PhD thesis, Royal Holloway, University of London, England, July 2008.

[13] Jiqiang Lu. *Cryptanalysis of Reduced Versions of the HIGHT Block Cipher from CHES 2006*. In Kil-Hyun Nam and Gwangsoo Rhee (editors), *ICISC 2007*, volume 4817 of *Lecture Notes in Computer Science*, pages 11-26. Springer, 2007.

[14] Eli Biham, Alex Biryukov, and Adi Shamir. *Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials*. *Journal of Cryptology*, 18(4), pages 291-311, 2005.

[15] Martin Feldhofer, Johannes Wolkerstorfer, and Vincent Rijnmen. *AES Implementation on a Grain of Sand*. *IEE Preceedings on Information Security*, Volume 152, Issue 1, pages 13-20, 2005.

[16] Eli Biham, Alex Biryukov, and Adi Shamir. *Impossible Differential Attacks*, http://video.google.com/videoplay?docid=1283860161748060032, *CRYPTO 1998*, Rump Session, last visited on July 2009.

[17] National Institute of Standards and Technologies. *Skipjack and KEA Algorithm Specifications*, http://csrc.nist.gov/groups/ST/toolkit/documents/skipjack/skipjack.pdf, Version 2.0, 29 May 1998, last visited on July 2009.

[18] National Institute of Standards and Technologies. *Announcing Request for Candidate Algorithm Nominations for the Advanced Encryption Standard (AES)*, http://csrc.nist.gov/archive/aes/pre-round1/aes_9709.htm, 12 September 1997, last visited on July 2009.

[19] National Institute of Standards and Technologies. *Commerce Department Announces Winner of Global Information Security Competition*, http://www.nist.gov/public_affairs/releases/g00-176.htm, 2 October 2000, last visited on July 2009.

[20] Lars Ramkilde Knudsen. *DEAL - A 128-bit Block Cipher*, AES submission, http://www2.mat.dtu.dk/people/Lars.R.Knudsen/newblock.html, last visited on July 2009.

[21] Xuejia Lai, James L. Massey, and Sean Murphy. *Markov Ciphers and Differential Cryptanalysis*. In Donald W. Davies (editor), Advances in Cryptology, *EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 17-38. Springer, 1991.

[22] Eli Biham, Alex Biryukov, and Adi Shamir. *Miss in the Middle Attacks on IDEA, and Khufu*. In Kaisa Nyberg (editor), *FSE '99*, volume 1636 of *Lecture Notes in Computer Science*, pages 124-138. Springer, 1999.

[23] Ralph C. Merkle. *Fast Software Encryption Functions*. In Alfred Menezes and Scott A. Vanstone (editors), Advances in Cryptology, *CRYPTO'90*, volume 537 of *Lecture Notes in Computer Science*, pages 476-501. Springer, 1990.

[24] Onur Özen, Kerem Varıcı, Cihangir Tezcan and Çelebi Kocair. *Lightweight Block Ciphers Revisited: Cryptanalysis of Reduced Round PRESENT and HIGHT*. In Colin Boyd and Juan Gonzalez Nieto (editors), *ACISP 2009*, volume 5594 of *Lecture Notes in Computer Science*, pages 90-107. Springer, 2009.

[25] Eli Biham and Adi Shamir. *Differential Cryptanalysis of Data Encryption Standard*. Springer-Verlag, 1993.

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

[27] Kaisa Nyberg and Lars Ramkilde Knudsen. *Provable Security Against Differential Cryptanalysis*. In Ernest F. Brickell (editor), Advances in Cryptology, *CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 566-574. Springer, 1993.

[28] Lars Ramkilde Knudsen. *Cryptanalysis of LOKI91*. In Jennifer Seberry and Yuliang Zheng (editors), Advances in Cryptology, *AUSCRYPT '92*, volume 718 of *Lecture Notes in Computer Science*, pages 196-208. Springer-Verlag, 1993.

[29] Lawrence Brown, Josef Pieprzyk, and Jennifer Seberry. *LOKI - A Cryptographic Primitive for Authentication and Secrecy Applications*. In Jennifer Seberry and Josef Pieprzyk (editors), Advances in Cryptology, *AUSCRYPT '90*, volume 453 of *Lecture Notes in Computer Science*, pages 229-236. Springer, 1990.

[30] Lawrence Brown, Matthew Kwan, Josef Pieprzyk, and Jennifer Seberry. *Improving Resistance to Differential Cryptanalysis and the Redesign of LOKI*. In Hideki Imai, Ronald L. Rivest and Tsutomu Matsumoto (editors), Advances in Cryptology, *ASIACRYPT '91*, volume 739 of *Lecture Notes in Computer Science*, pages 36-50. Springer, 1993.

[31] Arthur Sorkin. *Lucifer, a Cryptographic Algorithm*. *Cryptologia*, Volume 8, No. 1, pages 22-41. January 1984.

[32] Bruce Schneier and John Kelsey. *Unbalanced Feistel Networks and Block-Cipher Design*. In Dieter Gollman (editor), *FSE '96*, volume 1039 of *Lecture Notes in Computer Science*, pages 121-144. Springer, 1996.

[33] Young-Il Lim, Je-Hoon Lee, Younggap You, and Kyoung-Rok Cho. *Implementation of HIGHT cryptic circuit for RFID tag*. In *IEICE Electronics Express*, Volume 8, No. 4, pages 180-186, 2009.

[34] Soeren Rinne, Thomas Eisenbarth, and Christof Paar. *Performance Analysis of Contemporary Light-Weight Block Ciphers on 8-bit Microcontrollers*. In *ECRYPT workshop SPEED - Software Performance Enhancement for Encryption and Decryption* booklet, pages 33-43, 2007.

[35] Lars Ramkilde Knudsen. *Truncated and Higher Order Differentials*. In Bart Preneel (editor), *FSE '94*, volume 1008 of *Lecture Notes in Computer Science*, pages 196-211. Springer-Verlag, 1995.

[36] Alex Biryukov and David Wagner. *Slide Attacks*. In Lars R. Knudsen (editor), *FSE '99*, volume 1636 of *Lecture Notes in Computer Science*, pages 245-259. Springer-Verlag, 1999.

[37] John Kelsey, Bruce Schneier, and David Wagner. Key-Schedule Cryptoanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES. In Neal Koblitz (editor), Advances in Cryptology, *CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 237-251. Springer-Verlag, 1996.

[38] David Wagner. *The Boomerang Attack*. In Lars R. Knudsen (editor), *FSE '99*, volume 1636 of *Lecture Notes in Computer Science*, pages 156-170, Springer, 1999.

[39] Eli Biham, Orr Dunkelman, and Nathan Keller. *The Rectangle Attack - Rectangling the Serpent*. In Birgit Pfitzmann (editor), Advances in Cryptology, *EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 340-357. Springer-Verlag, 2001.