View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Concordia University Research Repository

APPLICATION OF CRYPTANALYSIS METHODS TO SOME SYMMETRIC KEY PRIMITIVES

ANAHID KHAJOOEIZADEH

A THESIS

IN

THE CONCORDIA INSTITUTE FOR INFORMATION SYSTEMS ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF APPLIED SCIENCE IN INFORMATION SYSTEMS SECURITY

CONCORDIA UNIVERSITY

MONTRÉAL, QUÉBEC, CANADA

APRIL 2015

© ANAHID KHAJOOEIZADEH, 2015

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: Anahid Khajooeizadeh

Entitled: APPLICATION OF CRYPTANALYSIS METHODS TO SOME SYMMETRIC KEY PRIMITIVES

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Ben Hamza Chair

Dr. Bentahar Examiner

Dr. Tarek Zayed Examiner

Dr. Amr Youssef Supervisor

Approved by _____
Chair of Department or Graduate Program Director

_____
Dean of Faculty

Date May 15, 2015

ABSTRACT

Application of cryptanalysis methods to some symmetric key primitives

Anahid Khajooeizadeh

Block ciphers and hash functions are important cryptographic primitives that are used to secure the exchange of critical information. With the continuous increase in computational power available to attackers, information security systems including their underlying primitives need continuous improvements. Various cryptanalysis methods are used to examine the strength and weakness of hash functions and block ciphers.

In this work, we study the Lesamnta-512 and DHA-256 hash functions and the LAC authenticated encryption scheme. In particular, we study the resistance of the underlying block cipher of the Lesamnta-512 hash function against impossible differential attacks and the resistance of the DHA-256 compression function against collision attacks. We also study MAC forgery attacks against LAC. Throughout our analysis, we use different automated methods to facilitate our analysis. For the cryptanalysis of Lesamnta-512, two automated methods are studied for finding an impossible differential path with the maximum length. Using the obtained impossible differential path, impossible differential cryptanalysis of Lesamnta-512 is performed for 16 rounds. For the DHA-256 hash function, we used an algebraic method to find collisions for its 17-step reduced compression function

by deriving difference equations for each step and then solving them when the conditions for collisions are imposed on these equations. For LAC, the differential behavior of the different operations of the cipher is represented into a set of linear equations. Then, a Mixed Integer Linear Programming (MILP) approach is used to find a high probability characteristic. This characteristic is then used to perform a forgery attack on LAC.

Acknowledgments

I would never have been able to finish this work without the guidance of my thesis advisor. I would like to express my deepest gratitude to him for his excellent guidance, caring, patience, and providing me with an excellent atmosphere for doing research.

I would also like to thank my parents, my sister and my brother. They were always supporting me and encouraging me with their best wishes. It was under their watchful eye that I gained so much drive and an ability to tackle challenges head on.

I would like to thank my love and best friend for his faith in me and allowing me to be as ambitious as I wanted. He always gives me the energy and confidence to work harder. He was always there cheering me up and stood by me through the good times and bad.

I would also like to thank my colleagues for sharing their experiences in this field and practical issues beyond the textbooks with patience.

My research would not have been possible without these helps.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

With the heavy reliance of our society on computers and the rising popularity of the Internet, the importance of data validation and authentication is on the rise. Cryptographic hash functions play a great role in securing modern information and computing systems. They are used as building blocks to achieve data integrity, authentication and non-repudiation as part of the digital signature schemes, message authentication codes and many other security schemes. On one hand, the steady increase in computational power allows us to deploy hash functions, among other cryptographic primitives, in many security applications. On the other hand, it also allows hackers to brute force weaker primitives and practically execute complex cryptanalytic techniques that were thought of as impractical in the past. In 2005, two commonly used hash functions were attacked successfully. Wang *et al.* released details on how to break the widely used MD5 hash function and presented examples for

colliding messages [29]. Afterwards, a theoretical attack was presented on SHA-1. This attack can find colliding pairs with complexity much less than predicted by its designed security parameters [28]. As a result, a continuous, and indeed challenging, development of hash functions is required in order to design more secure hash functions and be able to compute more precise security bounds. On November 2007, the National Institute of Standards and Technology (NIST) announced the SHA-3 competition in order to select a new hash function standard that addresses the issues in SHA-1 and MD5 [12]. DHA (Double Hash Algorithm)-256 hash function is one of the early designs introduced to strengthen the SHA-2 hash function. There are two main differences introduced in DHA compared to SHA-2. First, the message expansion formula is different and it has been changed in a way to use the nearest previous messages in its iterations rather than the far steps. This helps the message expansion to use more messages from previous steps and therefore makes the search for collisions harder. Lesamnta is another new family of hash functions submitted to the NIST SHA-3 competition. The Lesamnta hash function family includes four algorithms: Lesamnta-224/256/384/512. For each algorithm, the Merkle-Damgard domain extension with an output function is adopted [9]. Since the inner functions of Lesamnta are similar to the Advanced Encryption Standard (AES) [10], many optimized software and hardware implementation techniques for AES are also applicable to Lesamnta.

Poor composition of encryption schemes and authentication codes (MAC) can lead to dramatic security failure of the underlying security systems even when each of these primitives is secure by itself [3], [19], [27], [17]. Authenticated encryption schemes are primitives which simultaneously provide confidentiality, integrity, and authenticity assurances

on the data; decryption is combined in single step with integrity validation. These attributes are provided under a single, easy to use interface. Some applications require us to be able to securely use encryption schemes and authentication schemes together and provide authenticity and confidentiality for both secret and non-secret data (commonly referred to as the associated data). Thus authenticated encryption schemes with associated data were introduced to fulfil this requirement [26]. Based on the importance of the composition modes on the security of the information systems and renewed interest in the design and cryptanalysis of secure authenticated encryption schemes after the breaches on their plain combination, NIST funded the CAESAR competition (Competition for Authenticated Encryption: Security, Applicability, and Robustness) to specifically provide the security community with candidates for dedicated authenticated encryption primitives.

The LAC lightweight authentication encryption scheme is one of the candidates submitted to the CAESAR competition. The structure of LAC is based on ALE [7]. The primitive used in LAC as the back-end block cipher is a modified version of LBlock, called LBlock-s which is another lightweight block cipher.

In this thesis we study the three above mentioned cryptographic functions and schemes (Lesamnta-512, DHA-256 and LAC) and investigate three attacks on them. More precisely,

- We present some cryptanalytic results on DHA-256 and Lesamnta-512. In particular, we study the resistance of the underlying block cipher of the Lesamnta-512 hash function against impossible differential attacks, the resistance of DHA-256 compression function against collision attacks, and we study MAC forgery attacks against

LAC. Throughout our analysis, we use different semi-automated methods to facilitate our analysis. For the cryptanalysis of Lesamnta-512, two automated methods are studied for finding an impossible differential path with the maximum length. Using the obtained impossible differential path, impossible differential cryptanalysis of Lesamnta-512 is performed for 16 rounds.

- We study forgery attacks on LAC. In particular, the differential behaviour of the different operations of the cipher is represented into a set of linear equations. Then, a Mixed Integer Linear Programming (MILP) approach is used to find a high probability differential path. This path is then utilized to perform a forgery attack on LAC.

1.2 Contributions

Our contributions can be summarized as follows:

- We thoroughly study the Lesamnta-512 hash function and its underlying block cipher. Automated methods for finding the long impossible differential paths (the Matrix method and the Unified method) are studied. Based on our analysis, the Unified method turns out to be more suitable for application on Lesamnta-512. Using this method, a 16 round impossible differential path is found for the Lesamnta-512 block cipher.

- We present an algebraic attack to find collisions on reduced round version of the DHA-256 compression function.

- We present a forgery attack on the LAC authenticated encryption scheme. The attack was found by utilizing a Mixed Integer Linear programming approach to find a hight probability characteristic, and then we search for a high probability differential that follows the same set of active S-boxes within the found characteristic.

1.3 Thesis Organization

The rest of the thesis is organized as follows. In the next section, we present related work and some background that is required to understand the work presented throughout the thesis. In Chapter 3, Lesamnta-512 is studied where automated methods are used in order to find impossible differential paths with maximum length. Also, an algebraic attack to find collisions for a reduced round version of the DHA-256 compression function is presented. In Chapter 4, we present our forgery attack on the LAC authenticated encryption scheme. Finally, in Chapter 5 we present our conclusion and some possible future work directions.

Chapter 2

Related Work and Background

In this chapter, we briefly review some related work and provide background knowledge required to understand the work done in this thesis. First, we briefly review the security properties of cryptographic hash functions and authenticated encryption schemes. Then we provide background on differential cryptanalysis in general and impossible differential in particular. Finally, some automated methods that proved to be useful in the cryptanalysis of symmetric key primitives are described.

2.1 Properties of Cryptographic Hash Functions

2.1.1 Introduction

Hash functions are powerful tools in the design of techniques for protecting the authenticity of information. Typically, hash functions map larger domains to smaller ranges, referred to as digital fingerprints. This domain should be small enough in order to be able to efficiently

store it. In cryptography, hash functions are also used in authentication schemes where a hash function is combined with a secret key to form an authentication code (MAC). In Message Authentication Codes, the secret key is used in the compression process. Therefore it can be used to achieve data integrity (e.g., to ensure that the data has not been changed by an intruder or by a virus), and message authentication at the same time. The security of a MAC is dependent on the cryptographic strength of the underlying hash function and also on the size of its hash output and key. Additionally, hash functions are used in message fingerprinting, i.e., the hash value is used as the digital fingerprint of the message hashed since there is a low probability that two different messages hash to the same value. In case an intruder finds such messages, a collision is found and the hash function is broken. Another use of hash functions is in detecting any data corruptions in the received ciphertext. This is useful in verifying the integrity of files or messages and identifying them reliably. Hash functions are also widely used in digital signature schemes where a user attaches a code to their message as a signature. This is achieved by hashing the message and encrypting the produced hash with the user private key. Therefore the message can be verified and undeniable, i.e., the user who signed the message cannot deny this action. Also, hash functions are used in password verification schemes where it is not secure to save all the users' password or sensitive information in cleartext. This is achieved by storing the passwords' hash values instead and comparing the hash values in the verification process. In some proof-of-work systems, hash functions are used to verify that a work has been done by the user [13]. Other usages of hash functions are in pseudo random number generations and deriving keys from another key or password. For a cryptographic hash function, it is

assumed that the description of h must be publicly known and should not require any secret information for its operation. It is also assumed that the argument X can be of any arbitrary length and the result $h(X)$ has a fixed length of z bits. Furthermore, cryptographic hash functions should satisfy the following properties [24]:

- H is a hash function with z-bit output (z is a positive integer) if H is a deterministic function that takes an arbitrary length input and outputs a binary string of length z, i.e., $H : \{0,1\}^* \rightarrow \{0,1\}^z$.

- A hash function H is pre-image resistant (one-way) if, for a random string h, it is infeasible for an attacker, having h, to find m such that $H(m) = h$.

- A hash function H is second-preimage resistant if it is infeasible for an attacker having a random string m to find $m' \neq m$ such that $H(m) = H(m')$.

- A hash function H is collision resistant if it is infeasible for an attacker to find $m' \neq m$ such that $H(m) = H(m')$.

- The hash function must be collision resistant: this means that it is "hard" to find two distinct messages that hash to the same result.

Degree of Difficulty

If H maps the input vector A to output vector B then there are 2^z possible values for B considering that H has z-bits as output. The complexity of attacks is based on the number

of the hash function computations required to success. The complexities of generic pre-image, second pre-image, and collision attacks are as follows:

- Pre-image attack. In this attack, the attacker must find a message that produces a given a hash. Assuming the attacker uses an exhaustive search to find this message, the attack would have a complexity of 2^z.
- Second pre-image attack. The attacker must find another message that has the same hash value as a given message. Assuming the attacker uses an exhaustive search to find this message, the attack would have a complexity of 2^z.
- Collision attack. The adversary must find any two messages with the same hash value. Based on the birthday paradigm, the complexity of the collision finding attack is $2^{z/2}$.

2.2 The Birthday Paradigm

The birthday paradigm [24] states that in a random group of 23 people, the probability of finding two that share the same birthday is about 50%. In what follows, we sketch a simple proof for this paradigm. The chance that two people share the same birthday is $1/365$. The probability that two have different birthdays is $1 - 1/365 = 364/365$. Having z people, we conclude that we have $z(z-1)/2$ pair of people. Therefore, the probability that all these pairs have different birthdays is $(364/365)^{z(z-1)/2}$. The probability of finding at least one

pair that shares a birthday is given by

$$1 - (364/365)^{z(z-1)/2} \tag{1}$$

For $z = 1.2\sqrt{365}$ (253 pairs), Eq. 1 evaluates to 50%. This can be used for the hash function attack as well in order to find two messages hashing to the same value. Having a hash function H with z-bit output, then the complexity of finding collisions under this attack would be $z = 1.2\sqrt{2^z}$.

2.3 Cryptographic Properties of Authenticated Encryption Schemes

An important challenge in secure communication is preventing an unauthorized user from modifying the data going through the channel. Cryptography can be used to solve this problem. Generally, the authentication problem is different from the privacy problem, i.e., preventing an unauthorized user from extracting information from a communication channel. For entity authentication, it is important to identify the source and/or destination of a communication. Authenticated Encryption schemes are cryptographic primitives that are useful when confidentiality, integrity and origin authenticity of the data are required at the same time. Data integrity is not achieved with encryption only. Therefore, another authentication scheme is required to verify that the message has not been modified by an attacker using a code specific to the original user. This code is called a Message Authentication

Code (MAC). There are several cryptographic properties we would like to achieve in authentication encryption schemes. One property is indistinguishability. Assuming that the attacker has two messages and an encryption of one of them. If the attacker is not able to distinguish between them then the indistinguishability goal is met in that scheme. The weakest scenario is indistinguishability under chosen plaintext attack, called IND-CPA [4]. In this scenario the attacker has access to the scheme and therefore is able to encrypt, i.e., query this scheme with any two pairs of messages (m_0, m_1) and receive their encrypted value. The attacker wins if she is able to guess which one of the messages the encrypted value belongs to.

A stronger scenario is indistinguishability under chosen ciphertext attack, called IND-CCA [4]. It is also assumed that the attacker has repeated access to the encryption and decryption schemes. The attacker encrypts two messages (m_0, m_1) and receives an encrypted value. Then the attacker chooses a random ciphertext and guesses its corresponding plaintext value. The attacker wins if she is able to guess which one of the messages the encrypted value belongs to.

The two other security notions are integrity of plaintexts (INT-PTXT) and integrity of ciphertexts (INT-CTXT) [4]. In INT-PTXT, it should be computationally infeasible to find a ciphertext that when decrypted generates a plaintext that the sender has not encrypted. However, in INT-CTXT it should be computationally infeasible to find a ciphertext that has not been generated by the sender. INT-CTXT security implies INT-PTXT security.

There are three approaches to authenticated encryption: Encrypt-then-MAC (EtM) (encrypt first and then use the authentication key to compute the calculate the ciphertext tag),

Encrypt-and-MAC (E&M) (encrypt message and generate tag from the original plaintext) and MAC-then-Encrypt (MtE) (first the tag is calculated and both the message and tag are encrypted). In [19], it is shown that EtM is the secure approach compared to MtE and E&M.

There are some downsides to all these approaches. First, instead of one single block, two schemes are being used for authentication and encryption. Therefore not only this will have more complexity but also two different keys are required, one for each scheme meaning that the processing at the end user will be increased and this slows down the process. Additionally, just attaching and combining an encryption and an authentication scheme together will not necessarily provide a secure authenticated encryption scheme even if each scheme is proven to be secure [4] [19].

For these reasons, researchers are trying to design AE schemes that is not only efficient and secure but also uses the same key for both encryption and authentication. In order to do so, one approach is to introduce new modes of operation. These modes are designed using a block cipher and based on the assumption that the underlying block cipher is secure, i.e., it cannot be distinguished from a random permutation. For example, some ISO standard modes are Galois/Counter Mode (GCM) of operation, Counter with CBC-MAC (CCM) and Offset Codebook Mode (OCB) [2]. Another approach is to use a stream cipher and use a key for encryption and another for authentication, which can be generated by dividing the keystream into two parts [23].

The renewed interest in design and cryptanalysis of secure authenticated encryption

schemes, especially after the breaches caused by the use of their plain combination motivated NIST to fund the CAESAR competition (Competition for Authenticated Encryption: Security, Applicability, and Robustness) to specifically provide dedicated authenticated encryption schemes.

Generally, algorithms providing confidentiality and authenticity can be divided into two categories: authenticated encryption (AE) and authenticated encryption with associated data (AEAD). Each encrypts and authenticates plaintext data (in addition to authenticated-only data), which produces ciphertext with an authentication code. In the AEAD modes, separate data is authenticated, known as associated authenticated data or AAD. The associated authenticated data is only authenticated and not encrypted so it can be communicated unencrypted in cleartext. AEAD schemes are useful in applications where some information need to be authenticated but cannot be encrypted encrypted (the associated data in the AEAD mode). For instance the routing information in a packet should not be encrypted. However, it is vital to authenticate this information. On the other hand, the payload should be both encrypted and authenticated. Therefore the associated data would be the routing information in this mode. The second property of these schemes is that they should be able to process the data even without knowing the overall length of it. Another desirable property is parallelization, which allows the processing of different message blocks in a parallel manner.

The process for an authenticated encryption scheme can be summarized in the following:

- Encryption:

  - Input: The plaintext message M of arbitrary but usually limited length, a secret key K, a public message number (PMN) which is a publicly random number nonce and an optional associated data AD.
  - Output: A corresponding ciphertext C of the same length as the plaintext along with a fixed-length authentication tag T is generated.

- Decryption and Verification:
  - Input: The ciphertext C, the same secret key K, the same random nonce PMN that is used in the encryption process, the associated data AD if used in the encryption and the authentication tag T.
  - Output: Either the corresponding plaintext M or invalid ($\perp$) if the computed authentication tag $T\prime$ does not match the received one.

2.3.1 Security Requirements for Authenticated Encryption Schemes

An AEAD scheme should meet the following security requirements:

- Key Recovery: It should be infeasible to find the private key by any technique other than exhaustive search.
- Forgery Attack: It must be computationally infeasible to forge a valid tag for a chosen message if the private key is unknown. We call the attack an existential forgery attack if the attacker has chosen the plaintext and we call it a universal forgery attack if the message is given to the attacker.

- State Recovery: It should be computationally infeasible to recover the internal state of the AEAD primitive without knowledge of its key.

- Indistinguishability: It should be infeasible to distinguish the AEAD or AE primitive from a random function.

2.4 Differential Cryptanalysis

2.4.1 Introduction

In differential cryptanalysis, the general approach is to analyze the effect of modifying specific bits in two plaintext pairs on the differences in the resulting output ciphertext pairs. This process is performed on a set of plaintext and ciphertext pairs that have the same difference among each other. This is useful since we can compute the probabilities for each possible key and consequently find the most probable one. In this work, the exclusive-or operation (XOR) is used to calculate the difference between any two pairs. In the following sections we explain the basic ideas behind this attack in more details.

2.4.2 Basic Attack

We consider that our encryption system has the input (plaintext), $P = [P_1, P_2, ..., P_n]$, and output (ciphertext), $C = [C_1, C_2, ..., C_n]$. For two plaintext pairs $P\prime$ and P'', and their corresponding ciphertext pairs $C\prime$ and C'', the plaintext difference will be $\Delta P = [\Delta P_1, \Delta P_2, ..., \Delta P_n] = P\prime \oplus P''$. Our goal is to find the plaintext and ciphertext differences

that occur with high probability. This means that we are searching for a particular path (called a differential path) where a specific plaintext pair difference ΔP results in a specific ΔC with a high probability. The differential probability is determined by the differential distribution table of the underlying S-boxes as well as the number of active S-boxes. Let n_s denote the number of bits of the S-box input, we will have a $2^{n_S} \times 2^{n_S}$ matrix where the rows denote all possible input differences, the columns denote all possible output differences of the S-box and the cells denote the number of times that the corresponding input difference to an S-box will give the corresponding output difference. This table is called the difference distribution table and it shows the probability that a specific input difference to the S-box will result in a specific output difference. By propagating the plaintext pair difference forward in the encryption system and choosing the most probable input-output difference pairs, we can get a path that holds with a high probability and its differences in each round. Therefore we now have the input difference to the last round of the encryption system and also its output difference for the most probable differential path. In order to find the key, the attacker chooses the plaintext pairs such that they have the difference ΔP. Then, for all possible key values of the last subround, the attacker propagates from ΔC backwards and checks if the input difference to the last round is equal to the one for the most probable path. If it is equal then the counter for this key is incremented. Once all possible key values for the chosen set of plaintext and ciphertext pairs have been tested, the key with the highest counter value is chosen as the correct key. The impossible differential attack utilizes the probabilities in a different manner. In the following section, the impossible differential attack which is used in this work is explained in detail.

2.4.3 Impossible Differential Cryptanalysis

The impossible differential attack was first used in [5] to break the IDEA and Khufu block ciphers reduced to four and a half rounds and the Skipjack block cipher reduced to thirty-one out of thirty-two rounds. In this approach, instead of searching for the most probable paths in the cipher, we look for the paths whose probability of occurrence is exactly zero. These paths consist of the differences of two pairs of plaintexts, the difference of their internal values and their corresponding ciphertexts differences. We call a path an impossible differential path if the probability of its occurrence is zero. Thus while in the differential cryptanalysis [6] the differential characteristic (path) with the highest probability is exploited, in the impossible differential cryptanalysis [11] the approach is to look for the differential with zero probability, i.e., for plaintext/cipher differences that can never occur together. Since the correct key can never decrypt a ciphertext pair to that input difference, this information can be used to find the key. Therefore first an impossible differential path is found from ΔA to ΔB. A trial key that leads to contradictions is eliminated from the list of correct keys. By iterating over all possible last round key values and decrypting the ciphertext in the impossible differential path using that key, we can get the middle difference. Now if by encrypting the plaintext pairs driven from the impossible differential path using a trial key we get a contradicting difference in the middle, then we can omit this trial key since the probability that the path holds true for a correct key is zero. This way wrong keys are eliminated and once all possible key values for the chosen paths have been processed, we are left with a small candidate list of correct keys which has the correct key in it and sometimes left with a single correct key if adequate chosen pairs are used.

Chapter 3

Attacks on Lesamnta-512 and DHA-256

In this chapter, an impossible differential attack is applied to the underlying block cipher of the Lesamnta-512 hash function. We also provide a practical free-start collision for the DHA-256 compression function reduced to 17 steps using the algebraic approach proposed in [25].

3.1 Impossible Differential Cryptanalysis of Lesamnta-512

In this section, an impossible differential attack against the underlying block cipher of Lesamnta-512 hash function is presented. Two automated methods, namely, the Unified method [22] and the Matrix [18] method, were proposed in order to find impossible differential paths. These methods are first studied and analyzed for discovering the longest impossible differential path for our attack. The two techniques are compared in terms of their suitability for targeted block cipher structure. The results are then utilized to construct and

perform an impossible differential attack against Lesamnta-512. In the automated methods, the goal is to find the longest impossible differential path in a Block Cipher. This path is demonstrated in the form of $(\alpha_0, ..., \alpha_{n-1}) \not\rightarrow (\beta_0, ..., \beta_{n-1})$, i.e., the probability that the input difference $(\alpha_0, ..., \alpha_{n-1})$ leads to the output difference $(\beta_0, ..., \beta_{n-1})$ after r rounds is exactly zero. In these vectors, n denote the total number of sub-blocks of the considered block cipher and each element of the vector is its corresponding difference. A sub-block is defined as the group of bits that undergoes the same operations in the round function. Both the Matrix method [18] and the Unified method [22]. These techniques are automated methods that can be used to find the longest impossible differential path for any block cipher. In the following sections, first a background on the impossible differential cryptanalysis on Lesamnta-512 is given. Afterwards, the Matrix and the Unified methods are explained and then compared in terms of their performance and results. Based on the comparison results, the Unified method is determined as the suitable method for finding paths for the Lesamnta-512 block cipher. This method is modified and applied to Lesamnta-512 in order to perform an impossible differential cryptanalysis on it. Afterwards, the analysis of the impossible differential attack is given and finally this chapter is concluded.

3.1.1 Lesamnta-512 Preliminaries

In this work, we use the impossible differential attack to recover the key of the underlying block cipher of Lesamnta-512 hash function after the change in its round constants introduced to avoid the previous attacks performed on it [15].

The following notations will be used throughout this chapter: We denote the input

difference by α and the output difference after r-rounds is written as α^r. The ith sub-block of α (respectively α^r) is denoted by α_i (respectively ${\alpha_i}^r$). The input difference vector corresponding to α (respectively α^r) is denoted by $\vec{\alpha} = (\alpha_0, ..., \alpha_{n-1})$ (respectively $\vec{\alpha^r} = (\alpha_0^r, ..., \alpha_{n-1}^r)$). The ith entry of $\vec{\alpha}$ (respectively $\vec{\alpha^r}$) is denoted by α_i (respectively ${\alpha_i}^r$). In the decryption analysis we use the notations β, β_i, β^r, $\vec{\beta} = (\beta_0, ..., \beta_{n-1})$, β_i, $\vec{\beta^r}$ and ${\beta_i}^r$ instead of α, α_i, α^r, $\vec{\alpha}$, α_i, $\vec{\alpha^r}$ and ${\alpha_i}^r$ in the same order. Note that $\alpha = \alpha^0$, $\vec{\alpha} = \vec{\alpha^0}$, $\beta = \beta^0$ and $\vec{\beta} = \vec{\beta^0}$. Finding the longest impossible differential is the major step in attacking ciphers using impossible differential cryptanalysis.

Specfications of the Lesamnta-512 Block Cipher

The Lesamnta hash function family is categorized into three types based on their block sizes and data words used in their initial and the internal values [14]. These new family of hash functions have a narrow-pipe Merkle-Damgård structure and their algorithms are named Lesamnta-224 (256 bit block size), Lesamnta-256 (256 bit block size), Lesamnta-384 (512 bit block size and message digest size of 384 bits), and Lesamnta-512. Our focus in this work is Lesamnta-512 which has a block size of 512 bits long, a word size of 64 bits long and a message digest size of 512 bits long. Also, in this work the analysis is performed on the new version of Lesamnta-512 after introducing the change in its round constants. One of the main design criteria used in Lesamnta-512 has been its provable security using the Matyas-Meyer-Oseas (MMO) mode assuming that the underlying block cipher acts as an ideal primitive. Find a differential characteristic or impossible differential characteristic for the underlying cipher would imply that it is not ideal and renders the security proof for the hash function invalid. The Lesamnta-512 compression function consists of an internal

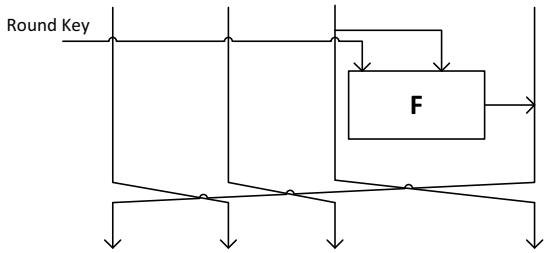

Figure 1: The internal block cipher structure of the Lesamnta-512 compression function

block cipher which is a generalized Feistel structure with 4 branches and has 32 rounds. Each branch is 128 bits long and the user key is 512 bits. The internal block cipher is utilized in the Matyas-Meyer-Oseas mode. Fig. 1 shows one round of the internal block cipher of Lesamnta-512. In the following section, the key scheduling algorithm is explained.

The Key Scheduling Algorithm of the Lesamnta-512 Block Cipher

The key scheduling algorithm for Lesamnta-512 block cipher is shown in Algorithm 1. In this algorithm, the input is the variable "chain" which is the 8 word input (each 64 bits). $Nr_{comp512}$ is the number of rounds of the block cipher which, for Lesamnta-512, is equal to 32 rounds. The variable K is a $Nr_{comp512} \times 2$ matrix with its rows being the round number and the columns being the first and second half of the round sub-key, i.e., the 128 bit subkey for round r will be the concatenation of $K[r][0]$ and $K[r][1]$. In each round, a round constant $C[round]$ (128-bit words) which is two 64 bits is added to $chain[4]$ and $chain[5]$. $C[round]$ is given by:

$$C[round] = 00000000000000XY0000000000000ZW \tag{2}$$

where XY is $2r + 1$ (in hex), and ZW is $2r$ for round r (in hex). The SubWords512, KeyLinear512, ByteTranspose512 and WordRotation512 functions used in the key scheduling algorithm are described below.

The SubWords512() function performs substitution in which it takes the input (128 bit in Lesamnta-512) and passes it from the Lesamnta-512 S-box and outputs the result. The KeyLinear512, ByteTranspose512 and WordRotation512 functions perform a permutation on their inputs. The input to KeyLinear512 and ByteTranspose512 are 16 bytes and for WordRotation512, the input is 8 words (each 64 bits). The specification of the permutation for KeyLinear512 function is shown in Eq. 4 and for ByteTranspose512, it is given by Eq. 3. Finally, WordRotation512 performs the permutation $x'_i \mod 8 = x_i$ on the input x_i where $0 \leq i \leq 7$.

Algorithm 1 The Key Scheduling Algorithm of Lesamnta-512

1: **procedure** KS(WORD $chain[8]$, WORD $K[Nr_{comp512}][2]$)
2: *word* $t[2]$;
3: *for* $(round = 0$ *to* $Nr_{comp512} - 1)\{$
4: $t[0] = chain[4] + C[round][0]$;
5: $t[1] = chain[5] + C[round][1]$;
6:
7: *SubWords512*(t);
8: *KeyLinear512*(t);
9:
10: *ByteTranspose512*(t);
11: $chain[6] = chain[6] + t[0]$;
12: $chain[7] = chain[7] + t[1]$;
13:
14: *WordRotation512*$(chain)$;
15:
16: $K[round][0] = chain[2]$;
17: $K[round][1] = chain[3]$;
18: $\}$

$$
\begin{aligned}
a'_0 = a_8, \quad a'_1 = a_9, \quad a'_2 = a_{10}, \quad a'_3 = a_{11}, \\
a'_4 = a_4, \quad a'_5 = a_5, \quad a'_6 = a_6, \quad a'_7 = a_7, \\
a'_8 = a_0, \quad a'_9 = a_1, \quad a'_{10} = a_2, \quad a'_{11} = a_3, \\
a'_{12} = a_{12}, \; a'_{13} = a_{13}, \quad a'_{14} = a_{14}, \; a'_{15} = a_{15}.
\end{aligned}
\tag{3}
$$

The KeyLinear512 permutation function equation is shown below. In this equation the multiplications are performed over $GF(2^8)$ field and $0 \leq i \leq 7$.

$$
\begin{bmatrix} a'_i \\ a'_{i+1} \\ a'_{i+2} \\ a'_{i+3} \\ a'_{i+4} \\ a'_{i+5} \\ a'_{i+6} \\ a'_{i+7} \end{bmatrix}
=
\begin{bmatrix}
01 & 01 & 02 & 0a & 09 & 08 & 01 & 04 \\
04 & 01 & 01 & 02 & 0a & 09 & 08 & 01 \\
01 & 04 & 01 & 01 & 02 & 0a & 09 & 08 \\
08 & 01 & 04 & 01 & 01 & 02 & 0a & 09 \\
0a & 09 & 08 & 01 & 04 & 01 & 01 & 02 \\
02 & 0a & 09 & 08 & 01 & 04 & 01 & 01 \\
01 & 02 & 0a & 09 & 08 & 01 & 04 & 01
\end{bmatrix}
\begin{bmatrix} a_i \\ a_{i+1} \\ a_{i+2} \\ a_{i+3} \\ a_{i+4} \\ a_{i+5} \\ a_{i+6} \\ a_{i+7} \end{bmatrix}
\tag{4}
$$

3.1.2 Methods for Finding Impossible Differential Paths with Maximum Length

3.1.2.1 The Matrix Method

In this section we describe the Matrix method introduced in [18]. This method is used to find the best impossible differential path in the terms of the path length for block ciphers based on the condition that the round function of the block cipher is a bijective function.

The Encryption and Decryption Characteristic Matrices

In this technique, two matrices are used in order to represent the encryption and decryption operations of a single round of the block cipher. One matrix is the encryption characteristic matrix, denoted as E, and the other is the decryption characteristic matrix, denoted as D. The dimension of these matrices is $n \times n$. The rows of these matrices denote the input differences corresponding to each sub-block which are a total of n sub-blocks and the columns denote the output differences corresponding to each sub-block (a total of n), for one single round of the block cipher.

Constructing the E Characteristics Matrix

The following principles are used in constructing the encryption characteristic matrix. The round function is denoted as F and its inverse is denoted as F^{-1}. The encryption characteristic matrix is constructed based on the following steps [18]:

1. If β_j is affected by α_i, i.e., $\beta_j = \alpha_i \oplus b$ where $\oplus$ is a modular addition and b is a

constant value, the entry $E(i, j)$ is set to 1.

2. If β_j is affected by $F(\alpha_i)$ or $F^{-1}(\alpha_i)$, the entry $E(i, j)$ is set to 1_F.

3. If β_j is not affected by α_j, the entry $E(i, j)$ is set to 0.

Constructing the D Characteristics Matrix

The following principles are used in constructing the decryption characteristic matrix [18]:

1. If α_j is affected by β_i, the entry $D(i, j)$ is set to 1.

2. If α_j is affected by $F(\beta_i)$ or $F^{-1}(\beta_i)$, the entry $D(i, j)$ is set to 1_F.

3. If α_j is not affected by β_j , the entry $D(i, j)$ is set to 0.

An example for the E and D matrices corresponding to the basic Feistel structure is shown in Fig. 2.

$$\mathbf{E} = \begin{bmatrix} 1_F & 1 \\ 1 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{D} = \begin{bmatrix} 0 & 1 \\ 1 & 1_F \end{bmatrix} \tag{5}$$

Definition 1 *Property one matrix: The E and D matrices are called property-one matrices if and only if the total number of 1 entries in each column is at most one. This automated technique requires both E and D to be property-one matrices.*

The Matrix Technique Process

In the first step we should analyze the propagation of the input differences through the block

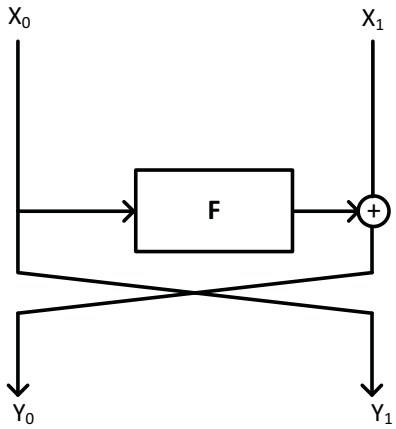

Figure 2: One round of the Feistel Structure

cipher after various rounds. We perform this analysis using the E and D characteristic matrices. However, first we categorize the values that the entries of both the input and output difference vectors can take. In this technique, the elements of the input and output difference vectors are not set to a specific input differences, i.e., all the differences that are useful in determining the best impossible differential path are considered. Therefore, for a given input difference, the possible output difference of each sub-block can belong to one of the following categories:

1. A zero difference (denoted as 0).

2. A non-zero non-fixed difference (denoted as 1 and δ).

3. A non-zero fixed difference (denoted as 1^* and γ).

4. Exclusive-or (XOR) of a non-zero fixed difference and a non-zero non-fixed difference (denoted as 2^* and $\gamma \oplus \delta$).

5. A non-fixed difference (denoted as t where $t \geq 2$).

In all the categories of differences above, a fixed difference is a difference that has not been affected by any round function F and a non-fixed difference represents a difference that has been affected by at least one round function F.

Now let us consider all possible values for the entries of the difference vectors $\vec{\alpha}$ and $\vec{\beta}$. The difference values of the initial input vector α^0 can only have fixed values since its sub-blocks are not affected by any round function F or F^{-1}. Therefore, we can conclude that the possible values of its elements can only be 0 or 1^*. For the encryption process, we need to know the output difference values of the difference vector $\vec{\alpha}$ after r rounds, that is $\vec{\alpha^r}$. Equivalently, it would mean that $\vec{\alpha^r}$ is the propagation of $\vec{\alpha}$ after the effect of the E matrix in r rounds. Similarly by using $\vec{\beta}$, $\vec{\beta^r}$ and the matrix D instead of $\vec{\alpha}$, $\vec{\alpha^r}$ and the matrix E, we can calculate $\vec{\beta^r}$ in the decryption process. Therefore, the values of the vector $\vec{\alpha^r}$ in terms of the vector $\vec{\alpha}$ can be computed as follows:

$$\vec{\alpha^r} = ((((\vec{\alpha} \cdot E) \cdot E) \cdots) \cdot E) = ((((\vec{\alpha^1} \cdot E) \cdot E) \cdots) \cdot E) = \cdots = \vec{\alpha^{r-1}} \cdot E) \quad (6)$$

In order to calculate $\vec{\alpha^r}$ using Eq. 6, we need to define the multiplication of $\vec{\alpha}$ and the characteristic matrix E. Eq. 7 defines this multiplication where $r \geq 0$. Also the multiplication $a_i.E_{i,j}$ is defined in Table 1 and the addition of $a_i.E_{i,j}$ and $a_{i'}.E_{i',j}$ is defined in Table 2.

$$\vec{\alpha} \cdot E = (\alpha_i)_{1\times n} \cdot (E_{i,j})_{n\times n} = (\sigma_i \alpha_i \cdot E_{i,j})_{1\times n} \quad (7)$$

$\alpha_i.E_i$	Meaning
$k.0 = 0$	The output difference of the jth sub-block not affected by the input difference α_i.
$k.1 = k$	The output difference of the jth sub-block affected by the input difference α_i.
$0.1_F = 0$	For a zero difference, the output difference after F is also zero.
$1^*.1_F = 1$	For a difference γ the output difference after F is δ.
$1.1_F = 1$	For a difference δ,the output difference after F is δ'.
$2^*.1_F = 2$	For a difference $\gamma \oplus \delta$, the output difference after F is t.
$t.1_F = t$	For a difference t, the output difference after F is also t.

Table 1: Multiplication between the difference vector entry α_i and the matrix entry E_i (where $k \in \{0, 1, 1^*, 2^*, t\}$ and $t \geq 2$) [18].

Addition	Exclusive-or
$k.0 = 0$	The output of the jth sub-block is not affected by the input α_i.
$0 + k = k$	$0 \oplus \Delta = \Delta$
$1 + 1 = 2$	$\delta \oplus \delta' = ?$
$1 + 1^* = 2^*$	$\delta \oplus \gamma = \delta \oplus \gamma$
$1 + 2^* = 3$	$\delta \oplus (\delta' \oplus \gamma) = ?$
$1 + t = 1 + t$	$\delta \oplus ? = ?$
$1^* + t = 1 + t$	$\gamma \oplus ? = ?$
$2^* + t = 2 + t$	$(\gamma \oplus \delta) \oplus ? = ?$
$t + t' = t + t'$	$? \oplus ? = ?$

Table 2: Relation of addition and XOR where $k \in \{0, 1, 1^*, 2^*, t\}$ and $t \geq 2$, $t' \geq 2$ and Δ is the corresponding difference for k. The $?$ mark denotes the differences which we do not know their value [18].

$\vec{\beta}^j$ can be calculated in the same manner as $\vec{\alpha}^i$ using the vector $\vec{\beta}$ and the characteristic matrix D.

The difference $t \geq 2$ is not useful in finding the differential path since these differences are non-fixed differences and so they can be 0 or any other non-zero value. Therefore, these differences in the difference vector do not provide us with any additional information on its value and are rendered useless. The useful values for the differences will be put in the set u. In this case, $u = \{0, 1, 1^*, 2^*\}$ since we excluded the value t. For each $m \in u$, an auxiliary set $\bar{m}$ is defined and used in finding the best impossible differential path in terms of the path length. This set is shown in Table 3. Let $\vec{\alpha}_i^r \in 2^*$ and $\vec{\beta}_i^{r'} \in \bar{2^*}$. Using the sets u

m	Differences	$\bar{m}$	Differences
0	0	$\bar{0}$ = {1, 1^*}	δ or γ.
1	δ	$\bar{1}$ = {0}	0.
1^*	γ	$\bar{1^*}$ = {0, 1^*, 2^*}	0 $\gamma' \neq \gamma$ or $\gamma \oplus \delta$.
2^*	$\gamma \oplus \delta$	$\bar{2^*} = \{1^*\}$	γ.

Table 3: Differences corresponding to the entries $m \in u$ and their corresponding entry sets $\bar{m}$ [18].

and $\bar{m}$, there will be an $r + r'$ round impossible differential path from α to β. We further explain the process of finding an impossible differential path with the maximum length in the following section.

Calculating the Longest Impossible Differential Path

Let M denote the length of the longest impossible differential path for the target block cipher.

Definition 2 *Given an input difference vector $\vec{\alpha}$, an entry $m \in u$ and the set $\bar{m}$, we define and calculate the following variables in the encryption process.*

$$\begin{aligned} ME_i(\vec{\alpha}, m) &\equiv max_r\{r|{\alpha_i}^r = m\}, \\ ME_i(\vec{\alpha}, \bar{m}) &\equiv max_{l\in\bar{m}}\{ME_i(\vec{\alpha}, l)\}, \\ ME_i(m) &\equiv max_{\bar{\alpha}\neq 0}\{ME_i(\vec{\alpha}, m)\}, \\ ME_i(\bar{m}) &\equiv max_{\bar{\alpha}\neq 0}\{ME_i(\vec{\alpha}, \bar{m})\}. \end{aligned} \tag{8}$$

The same definition applies to the decryption process but the vector $\vec{\beta}$ and the characteristic matrix D will be used instead of the vector $\vec{\alpha}$ and the characteristic matrix E, respectively.

$M(\vec{\alpha}, \vec{\beta})$ are defined as follows:

$$M(\vec{\alpha}, \vec{\beta}) \equiv max_{i,m}\{ME_i(\vec{\alpha}, m) + MD_i(\vec{\beta}, \bar{m})\} = max_{i,m}\{ME_i(\vec{\alpha}, \bar{m}) + MD_i(\vec{\beta}, m)\} \tag{9}$$

When the round function of a cipher is bijective and the encryption and decryption characteristic matrices E and D are both one-property matrices, M can be calculated as follows:

$$M = \max_{\vec{\alpha}\neq 0, \vec{\beta}\neq 0} M(\vec{\alpha}, \vec{\beta}) = \max_{i,m}\{ME_i(m) + MD_i(\bar{m})\} = \max_{i,m}\{ME_i(\bar{m}) + MD_i(m)\} \tag{10}$$

The algorithm used to calculate M using Eq. 10 is explained in the following section.

Algorithm to Compute M

This algorithm is proposed to calculate the length M assuming that the block cipher round function is a bijective function and the encryption and decryption characteristic matrices are one-property matrices. The algorithm consists of the following five steps:

1. Form the encryption characteristic matrix E.

2. Compute the values of $ME_i(m)$.

3. Form the decryption characteristic matrix D.

4. Compute the values of $MD_i(\bar{m})$.

5. Output the length M by applying $ME_i(m)$ and $MD_i(\bar{m})$ into Eq. 10.

In the first step, two variables $e_{i,j}$ and $\tilde{e}_{i,j}$ are used to represent the entries of $E_{i,j}$. The value of $e_{i,j}$ is set to zero if $E_{i,j} = 0$ and set $e_{i,j}$ is set to 1 if $E_{i,j} = 1$ or if $E_{i,j} = 1_F$. Additionally, we set $\tilde{e}_{i,j} = 0$ if $E_{i,j} = 1$ and set $\tilde{e}_{i,j} = 1$ if $E_{i,j} = 0$ or if $E_{i,j} = 1_F$.

In the second step, in order to represent the ith entry of the difference vector $\vec{\alpha}^r$, a new variable $a^{r,i}$ is defined and is assigned the integer value of the ith entry of $\vec{\alpha}^r$ without the $*$ symbol.

Similarly, the variable $\hat{a}^{r,i}$ is introduced and is set to 0 if the entry does not have the $*$ symbol. Otherwise it is set to -1.

Complexity of the algorithm

In this algorithm, steps 2 and 4 dominate the running time in which the algorithm iterates over all possible input differences of the vectors $\vec{\alpha}$ and $\vec{\beta}$. Note that in these steps the number of the sub-blocks and the possible values for each sub-block determine the number of the iteration itself. Therefore we have 2^n iterations to compute $ME_i(m)$ and similarly 2^n iterations to compute $MD_i(m)$. Thus, the time complexity of the algorithm is bounded by $\mathcal{O}(2^n)$.

Another method used to find the longest impossible differential path is the Unified method introduced [22]. This method provides us with longer paths than the ones derived with the Matrix method. However, some restrictions do apply as the complexity of the Unified

method will be high for block ciphers with more sub-blocks as compared to the Matrix method. Further comparison is performed in the following sections. Considering that Lesamnta-512 has four sub-blocks and based on the comparison results explained further, it is concluded that the Unified method can be a good candidate method for finding the longest impossible differential path for Lesamnta-512. In the section below the Unified method is explained.

3.1.2.2 The Unified Method

The Unified impossible differential path finding method [22] (UID-method) also tries to find the longest impossible differential path for block cipher structures in an automatic manner. Luo *et al.* [22] explain how this technique finds paths longer than the ones found by the previous technique.

It is assumed that the round function of the block cipher is a bijective function. The possible values for the difference vector are categorized in a different approach than that of the Matrix method. The possible values for the input and output difference vector are categorized as follows in the Unified method:

1. A zero difference (denoted by 0).

2. A non-zero fixed difference (denoted as l_i).

3. A non-zero varied difference (can be any value except zero and is denoted as m_i).

4. A varied difference (can be any value and is denoted as r_i).

Definition 3 *Two differences vectors* $\vec{\alpha} = (\alpha_0, \alpha_1, \cdots, \alpha_{n-1})$ *and* $\vec{\beta} = (\beta_0, \beta_1, \cdots, \beta_{n-1})$ *are inconsistent if there exists a subset* $I \in \{0, 1, ..., n-1\}$ *such that the XOR of the differences in the subset are always unequal, i.e.,* $\oplus_{i \in I} \alpha_i \neq \oplus_{i \in I} \beta_i$.

For example, if $\vec{\alpha} = (l_1 \oplus m_1, 0)$ and $\vec{\beta} = (l_1, 0)$ where l_1 is a non-zero fixed difference and m_1 is a non-zero varied difference, then $\vec{\alpha}$ and $\vec{\beta}$ are inconsistent since $l_1 \oplus m_1$ cannot be equal to l_1 in any condition. After i rounds in the encryption process we can compute $\vec{\alpha}^i$ and after j rounds in the decryption process we can compute $\vec{\beta}^j$. If $\vec{\alpha}^i$ and $\vec{\beta}^j$ are inconsistent then we can conclude that an impossible differential path of length $i + j$ exists. In this technique we define three types of transformations:

1. Zero transformation denoted as 0: If the input difference is $x \in \{0, l_i, m_i, r_i\}$ and the output difference is 0.

2. Identical transformation denoted as 1: If the input difference is $x \in \{0, l_i, m_i, r_i\}$ and the output difference is x.

3. Non-linear bijective transformation denoted as F.

Transformation F can occur in the different scenarios illustrated below:

- If the input difference is 0 and also the output difference is 0.

- If the input difference is a non-zero fixed difference l_i and also the output difference is a new non-zero varied difference m_j.

- If the input difference is a non-zero varied difference m_i and also the output difference is a new non-zero varied difference m_j.

- Otherwise, the output difference will be a new varied difference r_j, i.e., it can be any value.

The construction of the encryption characteristic matrix E and decryption characteristic matrix D is same as the Matrix method. However, the Unified method can be used even for the block ciphers that their E and D characteristic matrices do not satisfy the one-property. In this technique, a new concept about E and D is introduced. This concept can be well illustrated with an example. Suppose $(\beta_1, \beta_2) = (F(\alpha_1 \oplus \alpha_2), F(\alpha_1 \oplus \alpha_2) \oplus \alpha_2)$. Then the encryption function can be divided into two functions. The first function is $(z_1, z_2) = (\alpha_1 \oplus \alpha_2, \alpha_2)$ and the second function is $(\beta_1, \beta_2) = (F(z_1), F(z_1) \oplus z_2)$. Consequently, the characteristic matrix is given by

$$\mathbf{E_1}.\mathbf{E_2} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} . \begin{bmatrix} 1_F & 1_F \\ 0 & 1 \end{bmatrix} \tag{11}$$

where E_1 and E_2 are the encryption characteristic matrices for the first and second function, respectively.

Searching for the Longest Impossible Differential Path in the Unified Method

In this section, the algorithm and the steps used to calculate the longest impossible differential path using the Unified method are explained. This algorithm iterates over all the possible combinations of the input and output difference vectors and then iterates over all the possible combinations of the number of rounds to find the longest path using Definition 3.

The input to this algorithm is the $n \times n$ encryption characteristic matrix E, the decryption characteristic matrix D and an integer which is the round number which starts with $r = 0$. The output to this algorithm is the computation result of the longest impossible differential path $\Delta_{in} \not\rightarrow \Delta_{out}$. The algorithm performs the following steps in order to find the longest impossible differential path:

step 1. For a difference vector pair $(\vec{\alpha}, \vec{\beta})$, find the maximum integer $m = i + j$ such that $\vec{\alpha}^i = ((((\vec{\alpha}.E).E)...).E)$ (this multiplication is performed i times) and $\vec{\beta}^j = ((((\vec{\beta}.D).D)...).D)$ (this multiplication is performed j times) are inconsistent. If $r \leq m$, let $r \leftarrow m$ and $(\Delta_{in}, \Delta_{out}) \leftarrow (\vec{\alpha}, \vec{\beta})$.

step 2. Repeat step 1 until all the possible input and output difference vectors $(\vec{\alpha}, \vec{\beta})$ are enumerated.

step 3. Return $\Delta_{in} \not\rightarrow \Delta_{out}$.

Complexity of the Algorithm

Step 1 dominates the running time of the algorithm, in which it iterates over all the possible input and output differences of the difference vectors $\vec{\alpha}$ and $\vec{\beta}$. The number of the sub-blocks and the possible values for each sub-block determine the total number of these iterations. Assume we have n sub-blocks and each sub-block can take values 0 or l_i where $i \in \{1, 2, \cdots, n\}$ then we can conclude that the time bound of the algorithm will be $O(((n+1)^n - 1)^2) = O(n^{2n})$.

Analysis and Comparison of the Two Methods

The algorithms mentioned in section 3.1.2 are used to evaluate the immunity of block ciphers against impossible differential cryptanalysis by trying to find the longest impossible differential trail. In this section we compare between these two methods.

Both methods eliminate the difficulties and errors resulting from manual analysis by using a completely automated technique. Since in the Matrix method the encryption and decryption characteristic matrices must have one-property, this method cannot be applied to block ciphers that do not satisfy this property. However, this condition does not need to be satisfied in the Unified method for finding the longest impossible differential trail, i.e., the Unified technique can find the longest impossible differential path even if the D and E characteristic matrices do not satisfy the one-property.

The Unified method can find trails longer than the Matrix method for the same block cipher given as their inputs. This occurs for two reasons. First, the Matrix method cannot determine some kinds of inconsistency. As a result, some longer impossible differentials cannot be found (the tool using the Unified method mentioned in Section 3.1.2 has found trails longer than the ones found by the tool using the Matrix method). Definition 3 on inconsistency used in the UID method is more categorized than the one used in the tool using the Matrix method meaning that in the Unified method, the non-zero differences assigned to each sub-block are categorized in more categories. As a result, for the same block cipher, it finds impossible differential trails longer than the one found by the other tool. Secondly, in the Matrix method, the differences used at the beginning are either zero (0) or nonzero fixed differences (1^*). Therefore, in this method, it is considered that all the

non-zero differences are equal. Consequently, some longer paths may not be found.

Another consideration is that in both the Matrix method and the Unified method we do not take the detailed structure of the block ciphers round function into account. Consequently, this will lead to losing some trails that may be longer than the ones found by these tools. Additionally, the sub-block length is another factor that the Matrix method does not use, i.e., the number of the bits in each sub-block does not alter the results for the same block cipher. This may lead to losing some information about the differences that can be helpful in finding a longer impossible differential path. The same consideration is valid for the Unified method except for the fact that the differentiation in the non-zero difference assignment to each sub-block helps in utilizing some information for the sub-block length.

As mentioned in section 3.1.2, the time complexity of the impossible differential path algorithm for the Matrix method is $O(2^n)$. Therefore, for any block cipher of n sub-blocks where $n \leq 32$ we will be able to find the best impossible differential trail using this technique. On the other hand, the time complexity for the longest impossible differential path algorithm in the Unified method is $O(n^{2n})$. This time complexity makes this technique not suitable for block ciphers having more than 8 sub-blocks such as advanced encryption standard (AES) cipher which has 16 sub-blocks.

All these limitations mentioned for the Matrix method restricts this method to be a generic technique but results in a lower time complexity than the Unified method. Additionally any tool using the Unified method is not generic in the sense of the time complexity. This time complexity will make this method impractical for many ciphers that have sub-blocks greater than 8. However, in our case, the number of sub-blocks is relatively small.

Therefore the Unified method will be the suitable automated tool since it provides us with a longer impossible differential path than those found by the the Matrix method and is also practical in terms of its time complexity [22]. Therefore for Lesamnta-512 with less than 16 sub-blocks we use the Unified method to reach longer paths than the Matrix method.

3.1.3 Application of the Unified Method to Lesamnta-512

The Lesamnta hash function, introduced by Hirose *et al.* in 2009 [14] was submitted to the NIST cryptographic competition for hash algorithms. The internal block-cipher utilized in the Lesamnta-512 compression function is a generalized Feistel structure with 4 branches and has 32 rounds. Each branch is 128 bits and the user key is 512 bits. The detailed description is provided in section 3.1.1 and the structure is shown in Fig. 1. Bouillaguet *et al.* proposed a full round distinguisher for the underlying block cipher in [8]. As a result, in [16] the authors have performed a security analysis of the compression function of Lesamnta and the impact of it on full Lesamnta and proposed a change in the round constants to avoid the attack proposed by Bouillaguet *et al.*. It is concluded that a change in the round constants is required in order for Lesamnta to be secure against first and second pre-image resistance and collision resistance attacks. Lesamnta is of interest since it has efficient implementations and one of the main focuses in its design has been provable security by using Matyas-Meyer-Oseas (MMO) mode with the assumption that the underlying block cipher is acting as an ideal primitive. In this work, we use the impossible differential attack to recover the Lesamnta block cipher key. We have exploited the impossible differential behavior of Lesamnta-512 block cipher reduced to 16 rounds in order to derive its

key.

Using the Unified method, a 15 round impossible differential path is found and then is used as a distinguisher from a random permutation. The encryption characteristic matrix E and the decryption characteristic matrix D for the round Function of Lesamnta-512 are given as:

$$\mathbf{E} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1_F & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{D} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1_F \end{bmatrix} \tag{12}$$

The 15 round impossible differential path found using the Unified method is from $(0, 0, 0, l_1)$ to $(0, 0, l_1, 0)$. The forward path driven from this input difference vector is shown in Table 4 and the backward path is shown in Table 5 driven from the output difference vector. These two paths have inconsistency in the round they meet. In these impossible paths l_i denotes a non-zero fixed difference where $l_i \neq l_{i\prime}$ and m_i is a non-zero non-fixed difference, i.e., it is an output of a bijective function or key addition where $m_i \neq m_{i\prime}$, and finally r_i is a zero or any difference where $r_i \neq r_{i\prime}$. In all these notations, i is just a unique index for the different variables. In the following section, the Lesamnta family block ciphers specification is explained before illustrating the impossible differential attack on Lesamnta-512 block cipher.

Round number	First branch	Second branch	Third branch	Fourth branch
1	0	0	0	l_1
2	l_1	0	0	0
3	0	l_1	0	0
4	0	0	l_1	0
5	m_{39}	0	0	l_1
6	l_1	m_{39}	0	0
7	0	l_1	m_{39}	0

Table 4: The impossible differential path found in the forward direction using the Unified method. Each branch in each round has either zero, non-zero fixed (l_i), non-zero non-fixed (m_i) or other varied (r_i) difference.

Round number	First branch	Second branch	Third branch	Fourth branch
7	m_{56}	$l_1 + m_{58}$	$m_{52} + r_{97}$	$m_{54} + r_{98}$
8	m_{54}	m_{56}	$l_1 + m_{58}$	$m_{52} + r_{97}$
9	m_{52}	m_{54}	m_{56}	$l_1 + m_{58}$
10	l_1	m_{52}	m_{54}	m_{56}
11	0	l_1	m_{52}	m_{54}
12	0	0	l_1	m_{52}
13	0	0	0	l_1
14	l_1	0	0	0
15	0	l_1	0	0
16	0	0	l_1	0

Table 5: The impossible differential path found in the backward direction. Each branch in each round has either zero, non-zero fixed (l_i), non-zero non-fixed (m_i) or other varied (r_i) difference.

3.1.4 The Impossible Differential Attack

The first goal in using the impossible differential path derived from the Unified method is to calculate the number of structures needed in order to make sure all wrong keys are eliminated and therefore we are left with only one correct sub-key. Starting with 2^s structures , we set the expected value for the wrong keys left to be approximately 2^{-14}. First we find all the possible plaintexts that will have the input and output impossible differential differences $(0, 0, 0, a)$ and $(0, 0, a, 0)$, respectively. The output difference at round seventeen can only be resulted from a difference at step 18 where only the first and last two words have differences. Therefore, we have $2^{3\times128}$ possible values for the first three words of the impossible differentials and 2^{128} values for each pair in the last word. We conclude that all possible plaintexts with this input difference (output difference in step 17) are $2^{5\times128}$ and the portion of these pairs that also hold in the output difference are $2^{5\times128} \times 2^{-2\times128} = 2^{3\times128}$. So all these $2^{3\times128}$ pairs will definitely remove all the wrong keys since these are all the possible pairs that hold in the impossible differential path and we know that the probability of this paths occurrence is exactly zero. Considering our input difference, we have 2^{255} pairs for each structure since the input difference is of the form $(0, 0, 0, a)$ and for each 2^{128} values in each pair we choose two of them each time and the total number of pairs per structure would be $N_p = \binom{2^{128}}{2} = \frac{(2^{128})\times(2^{128}-1)}{2} = 2^{255}$.

As we concluded all $2^{3\times128}$ pairs will definitely remove all the keys. Therefore, by each structure we can be assured that one wrong key will be removed (based on the number of pairs per structure). Let $P(T)$ denote the probability that a structure would remove a specific key. Therefore, it is inferred that $P(T) = \frac{1}{2^{128}} = 2^{-128}$. Now let $P(F)$ denote

the probability that a structure does not remove a wrong key. Then we can deduce that $P(F) = 1 - P(T) = 1 - 2^{-128}$.

Therefore, each plaintext pair that holds the output difference will not remove the wrong key with a probability of $P(F) = 1 - 2^{-128}$. We have 2^s such structures and for each we have $N_p = 2^{255}$ plaintext pairs and 2^{-256} portion of these plaintext pairs will hold true in the output difference. Therefore we have a total of $N_t = 2^s \times 2^{255} \times 2^{-256} = 2^{s-1}$ of such plaintext pairs. Let $P(St)$ denote the probability that none of the structures remove a wrong key. Based on $P(F)$ and N_t we conclude that the probability that none of the structures remove a wrong key will be $P(St) = P(F)^{N_t} = (1 - 2^{-128})^{2^s \times 2^{255} \times 2^{-256}} = (1 - 2^{-128})^{2^{s-1}}$ since for each possible plaintext pair filtered according to the output difference, there is a probability of $P(F)$ that none of their structures remove a wrong key from the list of all possible key values.

Let N_w denote the number of wrong keys left in the list of all possible values of the sub-key. We have 2^{128} values for the last round sub-key and therefore based on the probability $P(St)$ we can calculate the expectation of the number of wrong keys left $(Exp\{N_w\})$ as $2^{128} \times (1 - 2^{-128})^{2^{s-1}}$ since we have 2^{128} possible values for each sub-key to guess and for each of these keys we have a probability $P(St) = (1 - 2^{-128})^{2^{s-1}}$ that none of the structures will remove this key.

As stated in the attack, we need the expectation of the number of wrong keys left to be equal to 2^{-14} in order to assure that an adequate number of structures and accordingly plaintexts have been passed and tried. By solving this equation we find the number of structures to be 2^{134}. Now that we know the adequate number of structures in order to be

able to find the correct key, we can apply the impossible differential cryptanalysis.

We Start with 2^{134} structures each consisting of $N_p = 2^{255}$ plaintext pairs per structure. We pass a total of $2^{255} \times 2^{134}$ plaintext pairs and filter out the pairs that hold true in the output difference of $(b, 0, 0, a)$ in round eighteen since the output difference at round seventeen being $(0, 0, a, 0)$ can only be resulted from a difference at step eighteen where only the first and last two words have differences. We can do this step by using the corresponding ciphertexts and looking up at their differences. Next, for each possible value of the last round sub-key and for each of these plaintext pairs, we collect the corresponding ciphertext using the guessed key and compute the difference of round seventeen by decrypting the values in round eighteen. If the values match the impossible difference, then we are assured that the guessed key is the wrong key and should be eliminated from the list. Otherwise, we move to the next guessed key value and repeat these steps. Therefore, the data complexity of our attack is about $2^{134} \times 2^{255} = 2^{389}$. In order to find the key, we must have all the 8 words (64 bits each) denoted as $chain[i]^r$ in a specific round r for $1 \leq i \leq 8$. However, using the algorithm for finding the corresponding sub-key, the second and the third words ($chain[2]$ and $chain[3]$) are known in each rounds sub-key. If we use the algorithm for rounds 13 to 17, $chain[2]^r$ and $chain[3]^r$ are found for $13 \leq r \leq 17$. The objective is to find the 512 bit key given the sub-keys for rounds 13 to 17. We note that due to the word rotation step in the key scheduling algorithm, $chain[2]^{13} \parallel chain[3]^{13}$ are equal to $chain[4]^{14} \parallel chain[5]^{14}$. Therefore, in round 14 we have $chain[i]^{14}$ for $2 \leq i \leq 5$. Let us denote $chain[6]^{14} \parallel chain[7]^{14}$ as the unknown variable X. As a result, in round 15, $chain[6]^{15}$ and $chain[7]^{15}$ are evaluated as a function of X. In this round $chain[i]^{15}$ is

known for $2 \leq i \leq 7$ up to an unknown parameter X. Also in the next round, again because of the word rotation, all the known values are shifted to the right and by adding the new sub-key values found by the algorithm, $chain[i]^{16}$ are known for $0 \leq i \leq 7$ except for X. The only remaining unknown parameter to find is the parameter X. Note that in round 17, $chain[6]^{15}$ and $chain[7]^{15}$ are now shifted to $chain[2]^{17}$ and $chain[3]^{17}$. As a result, X can be found. According to the key schedule algorithm five sub-keys are required in order to obtain the 512 bit user key and therefore the data complexity would be 5×2^{389}, the memory complexity would be 5×2^{128} and the time complexity would be $2^{261} \times 5$ encryptions based on the pairs that have the desired difference in steps 17 to 18 ($2^{389} \times 2^{-256}$). For each of these pairs the encryptions are performed for all possible values of subkeys, i.e. 2^{128}. Therefore the total number of encryptions for finding a subkey would be $2^{389} \times 2^{-256} \times 2^{128} = 2^{261}$. These encryptions should be performed for 5 subkeys to find the user key(therefore, 5 rounds).

The pseudo code for finding all words in step 16 is given in Algorithm 2:

3.2 Collision on Step-reduced DHA-256 Hash Function

In 2012, Keccak was announced as the winner of the NIST hash function competition. However, it has been noted that SHA-2 can be a better alternative than Keccak in terms of widely use of general CPUs based on eBASH project (ECRYPT Benchmarking of All Submitted Hashes). Therefore new hash functions following the SHA-2 design strategy are still attractive for software applications. Double Hash Algorithm (DHA)-256 hash function

Algorithm 2 Finding key given sub-keys of rounds $13 \leq i \leq 17$

procedure FINDING THE X VARIABLE
    $t \leftarrow chain[2]^{13} \parallel chain[3]^{13}$
    $t \leftarrow$ SubWords512(t)
    $t \leftarrow$ KeyLinear512(t)
    $t \leftarrow$ ByteTranspos512(t)
    $X \leftarrow \left(chain[2]^{17} \parallel chain[3]^{17}\right) - t$
procedure FINDING THE KEY
    $chain[0]^{16} \parallel chain[1]^{16} \leftarrow$ *Bit Operation* $\left(chain[2]^{13}\right.$ and $\left.chain[3]^{13}\right) + X$
    $chain[4]^{16} \parallel chain[5]^{16} \leftarrow chain[2]^{15} \parallel chain[3]^{15}$
    $chain[6]^{16} \parallel chain[7]^{16} \leftarrow$ *Bit Operation* $\left(chain[2]^{14} \parallel chain[3]^{14}\right) + chain[0]^{16} \parallel chain[1]^{16}$
procedure BIT OPERATION($INPUT : chain[2]^{13} \parallel chain[3]^{13}$, $OUTPUT : t$)
    $t \leftarrow chain[2]^{13} \parallel chain[3]^{13}$
    $t \leftarrow$ SubWords512(t)
    $t \leftarrow$ KeyLinear512(t)
    $t \leftarrow$ ByteTranspos512(t)

is one of the early designs introduced to strengthen the SHA-2 hash function. There are two main differences introduced compared to SHA-2. First, the message expansion formula is different and it has been changed in a way to use the nearest previous messages in its iterations rather than the far steps. This allows the message expansion to use more messages from previous steps and therefore makes the search for collisions harder. Initially, IAIK Krypto Group analyzed DHA-256 hash function and found a differential pattern which holds with a probability 2^{-63}, which is higher than assumed by the designers. Previous attacks on DHA-256 are pseudo-preimage and preimage on 35 step reduced DHA-256 [31]. However, the complexity is 2^{240}. In [20] a 9-step collision on DHA-256 is introduced with the complexity of 2^{-36}.

3.2.1 DHA-256 Preliminaries

In this work, using an algebraic method [25], a 17 step free-start collision is found on DHA-256 compression function. In particular, first an initial differential is derived in accordance with the message expansion and then a system of equations is determined based on the initial difference. By limiting the different conditions on the system of equations based on the initial differential table, the answers to the system are inferred and therefore the register values (chaining variables) are derived and a free-start collision is found. Hence, the problem of finding collisions is reduced to solving the system of equations and checking if there exists any answers for the message and register values that fit the conditions of our initial differential pattern instead of searching for all their possible values randomly.

3.2.2 Description of DHA-256

The DHA-256 hash function has message length of 512 bits and output length of 256 bits. There are 64 steps of operation. Messages are expanded for all the steps based on the message expansion algorithm in Eq. 13 and 14. For $0 \leq i \leq 15$, we have:

$$W_i = m_i \tag{13}$$

for $i > 15$ we have:

$$W_i = \sigma_1(W_{i-1}) + W_{i-9} + \sigma_2(W_{i-15}) + W_{i-16} \tag{14}$$

As seen in Fig. 3, in every internal state of DHA-256 compression function, we have 32-bit registers A, B, C, D, E, F, G and H which are called the chaining values. At the first step, these values are initialized with a fixed value and are updated for each step based on Eq. 15:

$$\begin{aligned}
A_{i+1} &= B_i, \\
B_{i+1} &= S_1(C_i), \\
C_{i+1} &= D_i, \\
D_{i+1} &= E_i + SS_2(H_i) + g(F_i, G_i, H_i) + W_i + K_i, \\
E_{i+1} &= F_i, \\
F_{i+1} &= S_2(G_i), \\
G_{i+1} &= H_i, \\
H_{i+1} &= A_i + SS_1(D_i) + f(B_i, C_i, D_i) + W_i + K_i.
\end{aligned} \tag{15}$$

Throughout this work, $\Delta SS_1(D_i)$ and $\Delta SS_2(H_i)$ are calculated as follows:

$$\begin{aligned}
\Delta SS_1(D_i) &= SS_1(D_i') - SS_1(D_i) \\
\Delta SS_2(H_i) &= SS_2(H_i') - SS_2(H_i)
\end{aligned} \tag{16}$$

3.2.3 The General Idea of Creating the Collisions

In this section, the technique to find the collisions is explained. First a differential path is derived. However, the message differences are not fixed and we assume that they take the value δ_i and then later on we try to find this value such that our differential path holds

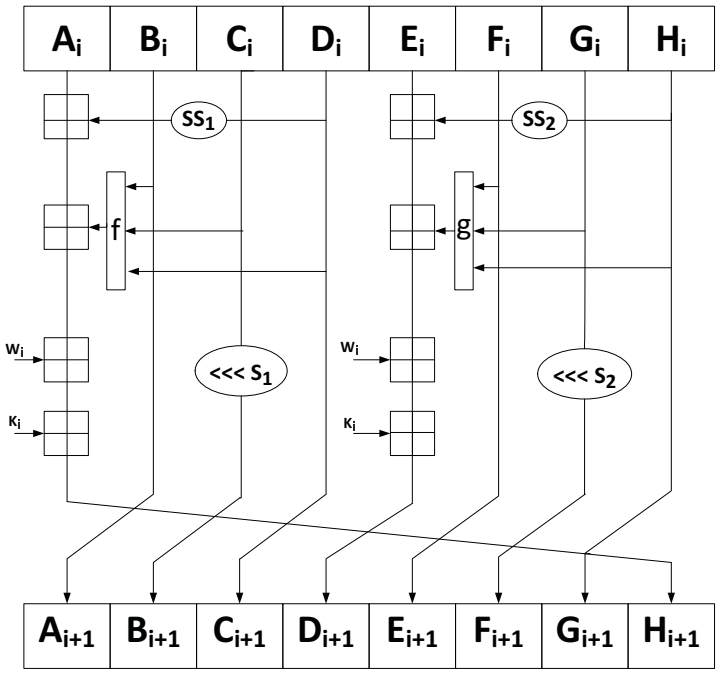

Figure 3: DHA-256 hash function structure

for all the step conditions. There are two prominent factors to consider for deriving this differential path. First, the message differences, δ_i's are introduced in order to cancel out some propagated differences. Hence, we need to assure there exists a value δ_i in step i such that the conditions for step i holds. Second, we need to consider the message expansion of the DHA hash function to assure the message differences introduced will not be used twice or more in our collision. Otherwise, this difference will be propagated again to further steps and the collision cannot occur. Assuming a difference is produced at step i of the hash function, we are going to create a differential path such that after some rounds of propagation, the register and the message differentials become zero. The path derived as shown in Table 6. As shown in the table, after the fourth step, all the differences in the registers and the message difference have been zero. Now our goal is to find values for δ_1 and δ_2 such that this differential path holds.

Step	ΔA	ΔB	ΔC	ΔD	ΔE	ΔF	ΔG	ΔH	ΔW
i	0	0	0	0	0	0	0	0	1
i+1	0	0	0	1	0	0	0	1	δ_1
i+2	0	0	1	0	0	0	1	0	0
i+3	0	1	0	0	0	1	0	0	0
i+4	1	0	0	0	1	0	0	0	δ_2
i+5	0	0	0	0	0	0	0	0	0

Table 6: The differential path derived for DHA-256.

3.2.4 Deriving the Equations for the Collision Conditions

In this section, difference equations are derived for each step and then the conditions are achieved based on them.

Equations and Conditions for Step $i+1$

In this step of the hash function, considering Table 6, we have $\Delta A_i = 0$, $\Delta E_i = 0$ and $\Delta W_i = 1$. We also know that we require ΔD_{i+1} and ΔH_{i+1} to be one since we want the message difference introduced in step i to propagate. Based on Eq. 15, we have:

$$W_i = D_{i+1} - E_i - SS_2(H_i) - g(F_i, G_i, H_i) - W_i - K_i \tag{17}$$

Therefore, for the differences we have:

$$1 = 1 - \Delta SS_2(H_i) - \Delta g(F_i, G_i, H_i) \tag{18}$$

Similarly based on Eq. 15 we have:

$$W_i = H_{i+1} - A_i - SS_1(D_i) - f(B_i, C_i, D_i) - K_i \tag{19}$$

and we deduct:

$$1 = 1 - \Delta SS_1(D_i) - \Delta f(B_i, C_i, D_i) \tag{20}$$

Since $\Delta f(0,0,0)$ and $\Delta g(0,0,0)$ are zeroes, then as we assume that ΔW_i will be one.

Equations and Conditions for Step $i+2$

We want ΔH_{i+2} and ΔD_{i+2} to be zero. Therefore, based on Eq. 17 and 19 we have:

$$\Delta W_{i+1} = -\Delta SS_1(D_{i+1}) - \Delta f^{i+1}(0,0,1) \tag{21}$$

$$\Delta W_{i+1} = -\Delta SS_2(H_{i+1}) - \Delta g^{i+1}(0,0,1) \tag{22}$$

In this equation, first notice that both ΔH_{i+1} and ΔD_{i+1} are set to 1 and they are being rotated to different bit locations by functions $SS_1(x)$ and $SS_2(x)$. Therefore, we need to find values of H_{i+1} and H'_{i+1} such that when their values are shifted, they remain the same. This way we are sure that their propagation in the bits would affect the solution of our system of equations. The only two values that have this property are:

$$D_{i+1} = 0 \qquad D'_{i+1} = -1 \tag{23}$$

$$H_{i+1} = 0 \qquad H'_{i+1} = -1 \tag{24}$$

We have fixed some of the register values at this point. Additionally note that it is required

that:

$$\Delta f^{i+1}(0,0,1) = \Delta g^{i+1}(0,0,1) \tag{25}$$

We are going to consider this condition while choosing the register values. Eq. 25 holds with probability $\frac{1}{2}$.

Equations and Conditions for Step $i+3$

Here, both ΔE_{i+2} and ΔA_{i+2} are zeroes. Besides, we know that we want the message to have no difference at this step. Hence, One may write:

$$\begin{aligned} \Delta W_{i+2} &= -\Delta f^{i+2}(0,1,0), \\ \Delta W_{i+2} &= -\Delta g^{i+2}(0,1,0) \end{aligned} \tag{26}$$

Considering the fact that both ΔH_{i+3} and ΔD_{i+3} should be zero in order for the equations to hold, it can be inferred that:

$$\begin{aligned} -\Delta f^{i+2}(0,1,0) &= 0, \\ -\Delta g^{i+2}(0,1,0) &= 0 \end{aligned} \tag{27}$$

Now we can fix some register values to assure this equation will hold. From Eq. 27, we infer that $B_{i+2} = C_{i+2}$ and also from the second term, we have $F_{i+2} = H_{i+2}$. Notice that this equalities can be propagated throughout the compression function. For instance, from Eq. 15 and by noting that $B_{i+2} = C_{i+2}$, we can infer that $B_{i+2} = C_{i+2} = A_{i+1}$ as well.

Equations and Conditions for Step $i+4$

Based on the differential path, all differences ΔE_{i+3}, ΔA_{i+3}, ΔH_{i+4} and ΔD_{i+4} are zero. Knowing that we want ΔW_{i+3} to be zero, we can write:

$$\begin{aligned} \Delta W_{i+3} &= -\Delta f^{i+3}(1,0,0), \\ \Delta W_{i+3} &= -\Delta g^{i+3}(1,0,0) \end{aligned} \tag{28}$$

For Eq. 28 to hold, we infer $C_{i+3} = D_{i+3}$. We also need $G_{i+3} = H_{i+3}$ so that the conditions for the second term of Eq. 28 are met.

Equations and Conditions for Step $i+5$

In this step, it is required that all registers differences are zeroes, otherwise, we do not have a collision. Additionally ΔE_{i+4} and ΔA_{i+4} are both one. Therefore, based on Eq. 17 one can infer:

$$\Delta W_{i+4} = -1 \tag{29}$$

We need to check this result with Eq. 19 as well to make sure there are no conflicts. Note that using this equation we get the same result. Finally, from Eq. 21 and 29 we infer the values for δ_1 and δ_2, respectively, as follows:

$$\begin{aligned} \delta_1 &= -\Delta SS_1(D_{i+1}) - \Delta f^{i+1}(0,0,1), \\ \delta_2 &= -1 \end{aligned} \tag{30}$$

Based on the DHA-256 hash function message expansion, assuming we start the differential

M_0	00000000 $164A245D$ 00000000 $C6A93DA7$ 00000000 00000000 00000000 00000000 00000000 00000000 $BD75D069$ 00000000 $8EC8BB70$ 00000000 $4A3F0433$ 00000000
M'_0	00000000 $164A245D$ 00000000 $C6A93DA6$ 00000000 00000000 00000000 00000000 00000000 00000000 $BD75D06A$ 00000000 $8EC8BB6F$ 00000000 $4A3F0433$ 00000000
H	$FFFFFFFF$ $FFFFFFFF$ $FFFFFFFF$ 00000000 $FFFFFFFF$ $FFFFFFFF$ 00000000 $FFFFFFFF$

Table 7: The DHA-256 17 Step Collision found. In this table in each row, the message and hash vector starts from the left and finishes at the end of the second line.

path in Table 6 from step 5, we note that each message will be used only once until step 17

and hence we have a collision on step 17. The results are shown in Table 7.

Chapter 4

Cryptanalysis of the LAC Authenticated Encryption Cipher

LAC was introduced in the CAESAR competition as a lightweight authenticated encryption cipher. The structure of LAC is based on ALE [7] and the primitive used in it as the back-end block cipher is a modified version of LBlock [30] called LBlock-s which is another lightweight cipher. In [7], the authors show that the probability of any characteristic is bounded by 2^{-70} at most since the input-output difference transforms for the S-box have probability bounded by 2^{-2} and the minimum number of active S-boxes for differential paths is 35. However, the ciphers resistance against differential cryptanalysis can be at risk if there exists many characteristics with high probability for some chosen candidate differential paths, i.e., the security bound is decreased in this case and the cipher can become vulnerable to differential cryptanalysis. In this work, we present a cryptanalytic attack on the LAC authenticated encryption cipher by using this information to find characteristics

with high probabilities for some candidate paths having the minimum number of active S-boxes. In order to achieve this, truncated differentials with the highest probability are derived. The minimum number of active S-boxes for each differential path is derived by converting the LAC function operations into linear equations to represent their differential behavior and then using a Mixed Integer Linear programming (MILP) approach to bound the number of active S-boxes. Finally a collection of characteristics are selected based on their input and output difference Hamming weights, minimum number of active S-boxes for that path and their corresponding truncated differential probabilities. These are used to find the exact differentials which enables one to compare the lower bounds on the probability of the differentials. The differential paths found can be used in to launch a forgery attack on LAC. Given the authenticated encryption (C, T), and a message M, the goal of our attack is to forge a valid ciphertext (C', T) by using the differential characteristic that hold with a high probability. Since the new ciphertext is valid, it can be decrypted to a message M'. Based on our experimental results, the probability of finding such ciphertext is higher than the claimed value of 2^{-70}, leading to security concerns for LAC. The main objective of the following sections is to find the paths with highest probability. In Section 4.7, characteristic paths with probability of $\approx 2^{-61.51}$ are found. The LAC authenticated encryption cipher is explained in section 4.1. Afterwards, we show how the differential behavior of the LAC authentication encryption operations are transformed into linear equations using the MILP algorithm and then we show how they are used to find the minimum number of active S-boxes in section 4.2.

4.1 LAC Authenticated Encryption Cipher Specification

LAC is a lightweight authenticated encryption cipher and is one of the candidates in the CAESAR competition. It uses a structure similar to ALE and one of its primitive blocks is a modified version of the LBlock cipher, called the LBlock-s function. The master key, K, in the encryption and authentication algorithm is 80 bits long and the authentication tag τ size in the output is 64 bits. At most 2^{40} bits of data can be authenticated and/or encrypted by using a master key with the same value. LAC has a nonce-respecting structure where a 64 bit public message number PMN is used as nonce to encrypt the message data m into the ciphertext c. The message m and its corresponding ciphertext c have the same length. PMN is only used once for each master key in the encryption process. Finally an associated data α is used in LAC with a variable length. The same variables are used in the decryption process using the ciphertext c as the input and the message m as the output. For authentication, the message m is returned only if the tag τ is correct and a special symbol denoting the failure of authentication is sent otherwise. Four underlying blocks are used for LAC encryption and authentication. These are the key scheduling algorithm KS, a full 32 rounds LBlock-s E, a 16 round LBlock-s G and a 16 round LBlock-s G-leak with 48 bits of leaked data state. All these blocks use LBlock-s as their basic function. In the following sections, first LBlock-s is explained and then these functions are illustrated. The encryption and authentication procedure for LAC authenticated encryption cipher is depicted in Fig. 4. In the following section we explain the LBlock-s cipher specification used as the underlying block for LAC.

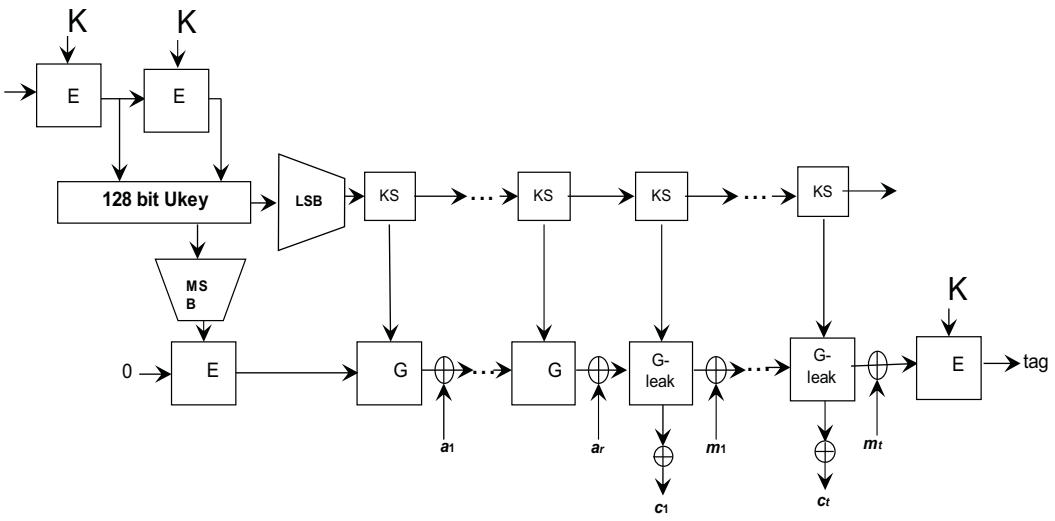

Figure 4: The encryption and authentication process of LAC authenticated encryption cipher

4.1.1 LBlock-s Specification

LBlock-s is a modified version of the LBlock lightweight block cipher with a variant of Feistel structure, 32 full rounds and a 64 bit data state as an input. The modifications are performed so that the implementation cost are reduced. Therefore, all rounds are identical in LBlock-s and there is no need to discard the switch operation in the last round of encryption since no decryption of this function is needed in LAC. This leads to a less cost of hardware control. Moreover, as opposed to LBlock which has 10 different S-boxes, LBlock-s has one 4-bit S-box. Another difference is the number of left bit shifts performed on the master key. In LBlock this number is 29 shifts which would be more costly to implement on 8 bit platforms whereas in LBlock-s, it is 24 bits which also improves the diffusion effect. The round function for LBlock-s has three steps: subkey addition, confusion function S and diffusion function P. In each round of LBlock-s, the 64 bit data state or input in

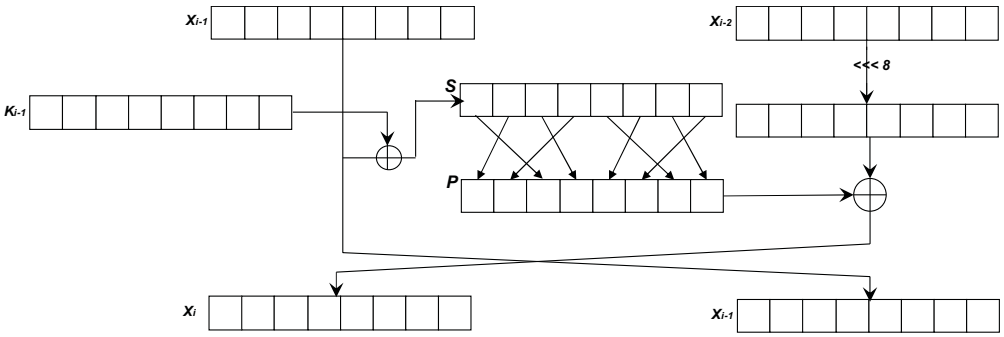

Figure 5: One round of LBlock-s cipher with its round function subkey addition, confusion and diffusion layers

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

Table 8: Input output pairs for the S-box used in LAC

round i is divided into two parts, X_{i-1} and X_{i-2}, and the round function is performed on these data states to get the data states X_i and X_{i-1} for the next round $i+1$ for $2 \leq i \leq 33$. Therefore, $X_{33}||X_{32}$ would be the output after full 32 rounds for the input $X_1||X_0$ where $a||b$ denotes the concatenation of the two binary strings, a and b. In the subkey addition step, each round subkey which is derived from the key scheduling algorithm explained in section 4.1.2 is XORed with half of the data state. The 4-bit S-boxes are used in the confusion step to provide data confusion by adding a non-linear layer. These parallel S-boxes are all identical in LBlock-s and the input-output pairs are shown in Table 8. Finally, a permutation is performed on eight nibbles in the diffusion step. Fig. 5 shows one round of LBlock-s and the details of the round function. Therefore, the round function can be described by Eq. 31 where $2 \leq i \leq 33$.

$$X_i = P(S(X_{i-1} \oplus K_{i-1})) \oplus (X_{i-2} <<< 8) \tag{31}$$

4.1.2 The Key Scheduling Procedure in LAC and LBlock-s

For each round i, the leftmost 32 bits of the master key K are used as the subkey for that round, $subkey_i$. The master key K is an 80 bit data, denoted $[k_{79}k_{78}...k_1k_0]$. Then the master key K is updated for the next round subkey using the following procedures:

$K <<< 24$, where $<<<$ denotes left bit shift with rotation.

$[k_{55}k_{54}k_{53}k_{52}] = s[k_{79}k_{78}k_{77}k_{76}] \oplus [k_{55}k_{54}k_{53}k_{52}]$

$[k_{31}k_{30}k_{29}k_{28}] = s[k_{75}k_{74}k_{73}k_{72}] \oplus [k_{31}k_{30}k_{29}k_{28}]$

$[k_{67}k_{66}k_{65}k_{64}] = [k_{71}k_{70}k_{69}k_{68}] \oplus [k_{67}k_{66}k_{65}k_{64}]$

$[k_{51}k_{50}k_{49}k_{48}] = [k_{11}k_{10}k_9k_8] \oplus [k_{51}k_{50}k_{49}k_{48}]$

$[k_{54}k_{53}k_{52}k_{51}k_{50}] = [k_{54}k_{53}k_{52}k_{51}k_{50}] \oplus [i]_2$

The round subkey $subkey_{i+1}$ is the leftmost 32 bits of the current master key K, where $[i]_2$ is the binary form of the round number i.

4.1.3 Specification of E, G and G-leak Functions

The E function used in the LAC authenticated encryption cipher is a full 32 round LBlock-s with $X_{33}||X_{32}$ acting as its output. However, the G function is a round reduced version of LBlock-s with 16 rounds of iteration in both the encryption procedure and the key scheduling algorithm. Therefore, the round equation for the G function would be as in Eq. 31 but with the condition that $2 \leq i \leq 17$ with $X_{17}||X_{16}$ as the output. Finally the G-leak

function is a 16 round LBlock-s with additional 48 bits of leaked internal data state. To derive these leaked bits, first, 9 rounds of LBlock-s is performed to get X_9 from $2 \leq i \leq 9$. The higher 24 bits of X_9 are saved as ${X_9}^*$ which is $X_9[31, 30, \cdots, 8]$ in bits. Then the next 8 rounds are performed from $10 \leq i \leq 17$ in order to obtain X_{17}. Similarly, the higher 24 bits of X_{17} are saved as ${X_{17}}^*$ which is $X_{17}[31, 30, \cdots, 8]$. Eventually, ${X_9}^*||{X_{17}}^*$ is saved as the leak bits as one output of this function where the data state $X_{17}||X_{16}$ acts as the other output. The key scheduling algorithm is also modified for the G-leak function. After deriving the subkeys for round numbers 2 to 16, using the same algorithm in KS function, another final round of key scheduling is performed from steps 1 to 7 but with the round constant $i = 0x15$. In the following section, the encryption procedure in LAC authenticated encryption cipher is explained.

4.1.4 Encryption and Decryption Procedures

The first step in the encryption process in LAC authenticated encryption cipher is the message padding. In this step, first the smallest number x of zeros is appended to the message m where len is the message length and the condition $(x + len + 40)$ mod $48 = 0; 0 \leq x \leq 48$ is met. Then the message length len coded on 40 bits is appended such that the length of the padded message is a multiple of 48 bits. Finally the padded message result is divided into t 48-bit blocks denoted as $M = m_1||m_2||\cdots||m_{t-1}||m_t$. Similarly, using the same steps, the associated data α is padded and is divided into r blocks, each having 48 bits. These blocks are denoted as $A = a_1||a_2||...||a_{r-1}||a_r$. The second step in the encryption process is the initialization, in which both the initial key state and the data state for LAC

are computed. As shown in Fig. 4, the public message number PMN (64 bits) and the master key K (80 bits) are given as the input to the E function and its 64 bit output is encrypted one more time using the master key K. An internal 128 bits key denoted as Ukey is constructed using the two 64-bit outputs in this step. The lower 80 least significant bits of the Ukey LSB are given as the initial key state to the KS functions and the higher 80 most significant bits of the Ukey MSB are given as the initial data state. Note that this initial data state is encrypted with 0 before being the input for the G functions. The third step in the encryption procedure is to process the associated data. In the first r rounds of LAC, the key state and the data state are given as the input to the G function and the lower 48 bits of output (internal data state) is added (exclusive-ored) to the first block (48 bits) of the associated data. The fourth step in the encryption process is processing the message. The output after the rth round along with the corresponding key state is used as the input to the G-leak function which gives two outputs. One is the leaked bits and the other is the internal data state. The internal data state is added to the lower 48 bits of the internal state to provide the input for the next G-leak function. The leaked bits which are the higher 24 bits of output after 8 rounds and the higher 24 bits of output after 16 rounds are added to the corresponding 48 message block denoted as m_i in order to compute the corresponding ciphertext block denoted as c_i in Fig. 4. The last step in the encryption process is finalization. The authentication tag for the message and the associated data τ is produced by encrypting the final data state using the master key K in function E. The decryption process is performed similarly with the exception that the ciphertext $c = c_1||\cdots||c_t$ is given as the input instead of the message m. Also the tag received will be compared to the tag computed

and, if not the same, an authentication failure symbol will be sent otherwise the decrypted message will be sent.

4.2 Finding Differential Paths with Minimum Number of Active S-boxes

In order to find the minimum number of active S-boxes, we need to solve an optimization problem by using the mixed integer linear programming technique explained in this section. Then the minimum number is used to find the candidate differential paths.

4.2.1 Mixed Integer Linear Programming

In order to compute the security bounds for the LAC authenticated encryption cipher against differential cryptanalysis (i.e., to analyze how plaintext differences propagate in ciphertext differences), a mixed integer linear programming method (MILP) is used in this work. First, we derive linear equations representing the differential behavior of the operations of the LAC authenticated encryption cipher given its structure as the input. The goal in linear programming (LP) is to optimize an objective function (also linear) having some conditions on the initial variables called the decision variables. We denote the linear objective function and decision variables as $f(x_1, x_2, ..., x_{v-1}, x_v)$ and x_i, respectively, where $1 \leq i \leq v$ and v denotes the total number of variables. The optimization in here is performed with the objective of minimizing the number of active S-boxes. We can use

this bound to find an upper bound for the probability of the best characteristic and therefore the best differential using the maximum differential probability (MDP) of the S-boxes. Therefore, when these equations, which define the differential behavior of each of the LAC authenticated encryption ciphers operations, are given as input to an MILP solver, we can get the minimum number of active S-boxes for LAC as the output. In this work the Ipsolver [1] has been used for solving the optimization problem. For candidate differential paths, the minimum number of active S-boxes for full 16 rounds of LAC authenticated encryption cipher has been calculated. Note that the decision variables x_i can have fixed integer values as conditions. The term MILP is used when some of the decision variables should be integer values and ILP term or pure linear programming is used when all of them should have integer values. In the following section, we demonstrate how MILP can be used to derive the minimum number of active S-boxes for a specific differential path.

4.3 Converting the Differential Behaviors of LAC Operations into Linear Equations

In this section we explain how the minimum number of active S-boxes is calculated using the MILP technique. First, the linear equations representing the differential behavior of the operations of the LAC authenticated encryption cipher are derived knowing that the G function in LAC is constructed using exclusive-or, rotational nibble shifts, confusion layers (S-box operations) and permutation layers. We denote the total number of input vector nibbles as nb. In the following section, we explain how the differential behavior of each of

these operations are converted into linear equations for the LAC authenticated encryption cipher. In order to do so, we use truncated differences as the input difference vector, i.e., having the difference vector $x = (x_0, x_1, \cdots, x_{nb})$, then $x_i = 0$ only if the corresponding nibble has the difference 0 as well. Otherwise the difference is one. Therefore, it is assumed that each nibble in the G function input can be either 0 or 1, i.e., each nibble has either a zero difference or a non-zero difference. These differences are called truncated differences. After deriving the linear equations for the G block of LAC authenticated encryption cipher, the objective function is determined; the objective function here would be the number of active S-boxes which is the summation of all the variables in equations that are given as an input to the confusion layer of the G function. Finally, to assure that the optimized result found from solving the equations does not contain the case where all the variables are zero and therefore the minimum number of active S-boxes is zero, we add an additional constraint that at least one S-box is active. We can achieve this by adding the constraint that the summation of all variables given as input to the confusion layer should be greater than or equal to one.

4.3.1 The Exclusive-or Operation Conversion

Let us assume the input difference vector entering the exclusive-or operation is $[{x_1}^{xor}, {x_2}^{xor}]$ and its corresponding output is ${x_{out}}^{xor}$. In this operation, the minimum number of input and output bytes that have differences is 2. We denote this number as the differential branch number. Note that the case where the input and output differences are all zero is not considered in finding this minimum. Therefore a dummy variable d^{xor} is defined to demonstrate

the differential branch number in the final linear equations corresponding to exclusive-or operation. The only case where this dummy variable d^{xor} is zero is when all variables ${x_1}^{xor}$, ${x_2}^{xor}$ and ${x_{out}}^{xor}$ are zero. The linear equations for the connection between input and output difference vector in exclusive-or operation are given by Eq. 32 where all these variables are binary variables.

$$\begin{aligned} &{x_1}^{xor} + {x_2}^{xor} + {x_{out}}^{xor} \geq 2d^{xor}, \\ &d^{xor} \geq {x_1}^{xor}, \\ &d^{xor} \geq {x_2}^{xor}, \\ &d^{xor} \geq {x_{out}}^{xor}. \end{aligned} \tag{32}$$

4.3.2 The Linear Operation Conversion

A linear operation is any operation that upon reception of an input vector $[{x^l}_1, {x^l}_2, \cdots, {x^l}_{nb-1}, {x^l}_{nb}]$, transforms it linearly to an output vector $[{x^l}_{out1}, {x^l}_{out2}, \cdots, {x^l}_{outn-1}, {x^l}_{outn}]$. Similar to the exclusive-or operation, the differential branch number B and the dummy variable d^l are used to demonstrate the connection between the differential input and output vectors. Likewise, only if all the variables ${x^l}_1,\ {x^l}_2,\ \cdots,\ {x^l}_{nb}, {x^l}_{out1},\ {x^l}_{out2}, \cdots,$ ${x^l}_{outn-1}$ and ${x^l}_{outn}$ are zero, then the dummy variable d^l would be zero. Otherwise, it will be one. The linear equations corresponding to the linear transformation operation are given

by Eq. 33.

$$
\begin{aligned}
& {x^l}_1 + {x^l}_2 + \cdots + {x^l}_{nb} + \\
& {x^l}_{out1} + {x^l}_{out2} + \cdots + {x^l}_{outn} \geq B d^l, \\
& d^l \geq {x^l}_1, \\
& d^l \geq {x^l}_2, \\
& \cdots \\
& d^l \geq {x^l}_{nb}, \\
& d^l \geq {x^l}_{out1}, \\
& d^l \geq {x^l}_{out2}, \\
& \cdots \\
& d^l \geq {x^l}_{outn}.
\end{aligned}
\tag{33}
$$

Note that finding the optimized result of the derived linear equations, we can restrict all dummy variables and input or output variables to be binary. In that case we need integer linear programming to find the optimized result. In the following section, we illustrate the construction of the MILP problem for the LAC authenticated encryption cipher, the calculation of the minimum number of active S-boxes for our differential path and how the differential cryptanalysis is performed using this technique.

4.4 Differential Cryptanalysis of LAC Authenticated Encryption Cipher Using the MILP Algorithm

In this section, we show how to construct the linear equations for the differential behavior of the LAC authenticated encryption cipher and then use these equations to find the minimum number of active S-boxes for several candidate differential paths. The variables used in the equations are in the form of mixed-integer equations, i.e., the decision variables are integer values. Therefore, this problem is an MILP problem and we use IPsolver to find the optimized result of the derived linear equations. The result of this step would be the minimization of the objective function which is the minimum number of active S-boxes for the given differential input vector to the specific output vector in the differential path for the 16 round LAC authenticated encryption cipher. This is used to perform differential cryptanalysis on the LAC authenticated encryption cipher by finding an actual characteristic with that given number of active S-boxes. Note that the S-box will output a non zero output difference only if the input difference is a non zero value. Therefore, we can map any active difference in the input to the output and keep the rest as a zero difference. The G function in LAC has one exclusive-or operation with inputs other than the subkey variables. Here we demonstrate the propagation of the difference vectors in LAC by forming the linear equations corresponding to these operations starting from the first round as an example. As depicted in Fig. 6, we denote the variables in the input difference vector by the binary values x_0 to x_{15}. Therefore, the initial left side input difference vector would be $[x_0, x_1 \cdots, x_6, x_7]$ and similarly the initial right side input difference vector would be

$[x_8, x_9, \cdots, x_{14}, x_{15}]$. The left side vector will be the next round right side vector and therefore the variables x_0 to x_7 will be on the right side for the next round. However, in order to construct next round left side vector, these variables should pass by the S-boxes. Since the S-box is bijective, it will transform any non-zero difference to another non-zero difference and hence every difference in the S-box input will be passed to the output and any zero difference will not. Next, these differences will be permuted and we apply these permutations in the S-box output as shown in Fig. 6. This vector will be used as one of the inputs for the exclusive-or operation of LAC authenticated encryption cipher which we will use in defining the linear equations. The right side vector corresponding to the variables x_8 to x_{15} will be shifted 8 bits to the left with rotation and the resulting vector is the second input to the exclusive-or operation of the LAC authenticated encryption cipher. Therefore the two vectors given as input to the exclusive-or operation in the first round of LAC are $[x_1, x_3, x_0, x_2, x_5, x_7, x_4, x_6]$ and $[x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_8, x_9]$ corresponding to the permutation output and 8 bit left rotation shift, respectively. We now define new binary variables x_{16} to x_{23} to denote the output corresponding to the input vectors for the exclusive-or operation. Note that these variables will be the left side vector for the next round since they are the result of the exclusive-or operation. By defining another new dummy variable, d_0, we can write down the linear equations for this round and for input

variables x_1 and x_{10} as follow:

$$\begin{aligned} x_1 + x_{10} + x_{16} &\geq 2d_0, \\ d_0 &\geq x_1, \\ d_0 &\geq x_{10}, \\ d_0 &\geq x_{16}. \end{aligned} \tag{34}$$

Similarly, considering the other exclusive-or operations in this round for other variables, we can get $8 \times 4 = 32$ equations for the first round since we get 4 equations for each exclusive-or operation and we have eight nibbles for the inputs to the exclusive-or operation. Therefore, for the new variables $x_{17}, x_{18}, x_{19}, x_{20}, x_{21}, x_{22}$ and x_{23}, similar equations with dummy variables $d_1, d_2, d_3, d_4, d_5, d_6, d_7$ are written and a system of 32 equations are used to describe all the operation for each round of the G function for LAC. This set of equations is given in the following form for round $r = 0$:

$$\begin{aligned} x_{k[i]} + x_{m[i]} + x_{n[i]} &\geq 2d_i, \\ d_i &\geq x_{k[i]}, \\ d_i &\geq x_{m[i]}, \\ d_i &\geq x_{n[i]}. \end{aligned} \tag{35}$$

for $i = 0, \ldots, 7$ where arrays $k = [1, 3, 0, 2, 5, 7, 4, 6]$, $m = [10, 11, 12, 13, 14, 15, 8, 9]$ and $n = [16, 17, 18, 19, 20, 21, 22, 23]$. As shown in Fig. 6, after one round, the propagated differences will be $[x_{16}, x_{17}, x_{18}, x_{19}, x_{20}, x_{21}, x_{22}, x_{23}]$ and $[x_0, x_1, \cdots, x_6, x_7]$ for the left

side and right side, respectively. The same process can be performed on every round of the LAC authenticated encryption cipher and therefore the whole structures differential behavior can be written as a system of linear equations with new defined binary variables. The overall set of conditions for rounds $r = 1, \ldots, 15$ can be written as:

$$\begin{aligned} & x_{k[i]+8+8r} + x_{m[i]-16+8r} + x_{n[i]+8r} \geq 2d_i, \\ & d_{i+8r} \geq x_{k[i]+8+8r}, \\ & d_{i+8r} \geq x_{m[i]-16+8r}, \\ & d_{i+8r} \geq x_{n[i]+8r}. \end{aligned} \tag{36}$$

for $i = 0, \ldots, 7$ where arrays $k = [1, 3, 0, 2, 5, 7, 4, 6]$, $m = [10, 11, 12, 13, 14, 15, 8, 9]$ and $n = [16, 17, 18, 19, 20, 21, 22, 23]$. This, along with Eq. (35), give a set of $16 \times 8 \times 4 = 512$ linear equations. In the following section, we describe how these linear equations are used in order to find the minimum number of active S-boxes for several candidate differential paths.

4.4.1 Finding the Minimum Number of Active S-boxes

In order to determine the minimum number of active S-boxes, we need to consider the variables in our linear inequalities that are given as input to the S-boxes in each round of the G function since these are the variables that determine if an S-box is active or not. For example, in round one, we can see that the variables $x_0, x_1, \cdots, x_6, x_7$ represent the S-box inputs so that these are the variables in the equations that should be considered in

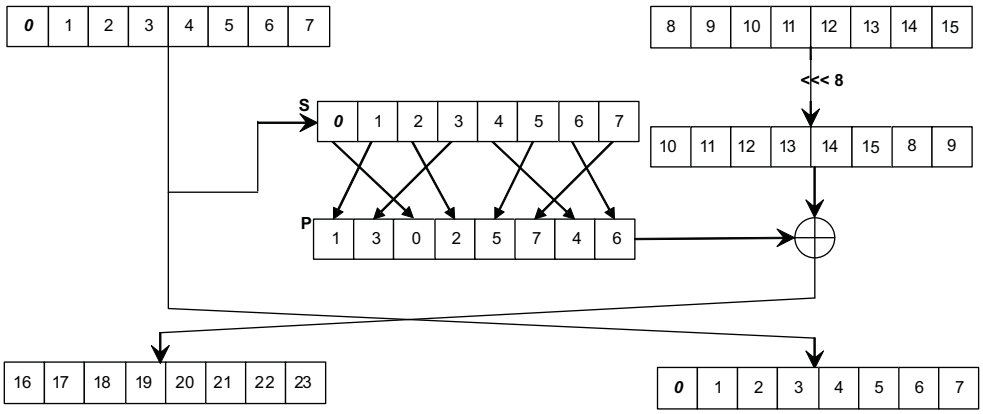

Figure 6: Differential propagation for the first step of LAC in initialization of the linear equations

our optimization problem. Note that we need to have an additional linear equation for the summation of these variables to be greater than or equal to one since we do not want the result where there is no active S-box. We denote the set of 8 variables determining the S-box inputs in round i as the set L_i and since the G function has 16 rounds, then we can conclude that $1 \leq i \leq 16$. Therefore, the sets for the LAC authenticated encryption cipher can be described as in Eq. 38. The number of active S-boxes k_N for a total of N rounds of the G function is given using Eq. 37, where $I_N = \bigcup_{i=1}^{N} L_i$.

$$k_N = \sum_{i \in I_N} x_i \tag{37}$$

Our objective would be to optimize, i.e., minimize, k_N. Having 32 linear equations in each round plus one additional equation for the S-box constraint for N rounds, we have a system of $32N + 1$ linear equations with binary variables (an ILP program). In this work, IPSolver is used for optimizing the equations according to the objective function. The minimum

number of active S-boxes for different candidate differential paths is compared and the results are shown in Table 9. In the following sections, we explain how these candidate paths were chosen for our experiment.

$$
\begin{aligned}
L_0 &= 0, 1, 2, 3, 4, 5, 6, 7, \\
L_1 &= 16, 17, 18, 19, 20, 21, 22, 23, \\
L_2 &= 24, 25, 26, 27, 28, 29, 30, 31, \\
L_3 &= 32, 33, 34, 35, 36, 37, 38, 39, \\
L_4 &= 40, 41, 42, 43, 44, 45, 46, 47, \\
L_5 &= 48, 49, 50, 51, 52, 53, 54, 55, \\
L_6 &= 56, 57, 58, 59, 60, 61, 62, 63, \\
L_7 &= 64, 65, 66, 67, 68, 69, 70, 71, \\
L_8 &= 72, 73, 74, 75, 76, 77, 78, 79, \\
L_9 &= 80, 81, 82, 83, 84, 85, 86, 87, \\
L_{10} &= 88, 89, 90, 91, 92, 93, 94, 95, \\
L_{11} &= 96, 97, 98, 99, 100, 101, 102, 103, \\
L_{12} &= 104, 105, 106, 107, 108, 109, 110, 111, \\
L_{13} &= 112, 113, 114, 115, 116, 117, 118, 119, \\
L_{14} &= 120, 121, 122, 123, 124, 125, 126, 127, \\
L_{15} &= 128, 129, 130, 131, 132, 133, 134, 135, \\
L_{16} &= 136, 137, 138, 139, 140, 141, 142, 143.
\end{aligned}
\tag{38}
$$

4.5 Choosing the Candidate Truncated Characteristics

To choose the candidate truncated characteristics, the following steps have been taken. First, we generate a matrix of the probabilities of all the possible truncated differentials of LAC over one round assuming its F-function is a random permutation. In truncated differentials, each nibble is either active (set to 1) or inactive (set to 0). Therefore, the size of this matrix is $2^{16} \times 2^{16}$. Then using MATLAB, we raised this matrix to the power of 16 so that we have the probabilities of the truncated differentials over 16 rounds. Afterwards, as LAC does not permit the manipulation of its whole 16-nibbles internal state, i.e., just 12 nibbles. We focused our attention on a truncated matrix of size $2^{12} \times 2^{12}$. In this truncated matrix, we searched for the entry that has the maximum probability corresponding to all possible input and output Hamming weights. Hence, after this step we ended up with 12×12 patterns corresponding to all the possible input and output Hamming weights and the maximum probability corresponding to each pattern. Using the MILP technique, we found the truncated characteristic corresponding to each of these 144 patterns and the number of active S-boxes for each truncated characteristic. The list of patterns was shorten by setting the number of active S-boxes to be 35, i.e., the minimum number of active S-boxes in 16 rounds. From the truncated difference matrix, we know if each nibble is active or not for the input pattern and the output pattern only and we do not know how this input difference pattern was propagated to reach the corresponding output difference pattern. This is what is called a differential, i.e., just the input difference and the output difference after some specific rounds. In a characteristic, we have the intermediate difference patterns

as well and that is what we get from the MILP and IP Solver. Then for this short list of truncated characteristics, we find the corresponding differential with the highest probability as explained in the next section.

4.6 Finding the Differential of the highest Probability

In this section, we explain the algorithm of turning the truncated characteristics we obtained in the previous section into differentials with specific input and output difference values. To speed up the algorithm, we restrict the number of active nibbles in each round. In particular, we restrict the input to have 3 active nibbles at most, the intermediate rounds to have 6 active nibbles in total and the number of active nibbles that pass by an S-box is limited to 3 out of these 6 active nibbles in each round. For a given truncated characteristic D, we have so many possibilities for the differences at the input, intermediate rounds and output. For a specific input/output difference pair, we consider all the possible differences in the intermediate round following our truncated characteristic D. This determines a group of characteristics that contribute to the same differential. Computing the sum of probabilities of those characteristics give a more accurate lower bound of the differential probability. We used the algorithm proposed in [21]. We denote a version of D with only i rounds as D_i. To compute $P(D : \alpha \longrightarrow \beta)$ for a given (α, β), we first compute the $P(D_i : \alpha \longrightarrow x)$ for all the differences x following D_1. Then $P(D_i : \alpha \longrightarrow x)$ is iteratively built for all x

$Truncated$ $Input$ $Difference$	$Truncated$ $Output$ $Difference$	$Probabilityof$ $Truncated$ $Differential$	$Numberof$ $Active$ $Sboxes$	$input$ $difference$ $value$	$output$ $difference$ $value$	$Probability$ of $Differential$
[0000000000001101]	[0000011000001100]	$2.75E-15$	35	[0000000000007604]	[0000044000002400]	$2^{-61.73}$
[0000000000001101]	[0000111000001100]	$4.12E-14$	35	[0000000000007604]	[0000442000004200]	$2^{-61.83}$
[0000101000000001]	[0000101100000000]	$1.83E-16$	35	[0000404000000004]	[0000706400000000]	$2^{-61.51}$

Table 9: The result characteristics with the minimum number of active S-boxes and the highest probability corresponding to the input and output difference values for the full 16 round of LAC authenticated encryption cipher

following D_i using the results from D_{i-1}:

$$P(D_i : \alpha \longrightarrow x) = \sum_{x'} P(D_{i-1} : \alpha \longrightarrow x') \times P(x' \longleftrightarrow x) \tag{39}$$

The results are presented in the following section.

4.7 Results

In this section, the result for each candidate truncated characteristic is shown in Table 9 in terms of the minimum number of active S-boxes and the probability of the truncated differential derived from the truncated difference table matrix. In this table, the probability of truncated differential is the probability from the truncated difference table matrix. The minimum number of active S-boxes corresponding to the truncated input difference and the truncated output difference is shown for the full 16 rounds of LAC, and probability of differential is the differential with the highest probability for the input difference and output difference, and finally the input and output difference values are the differentials determining the exact value of the nibbles.

Each digit in the truncated input difference shows whether that nibble is active or not. As an example $[0000000000101101]$ means the nibbles 10, 12, 13 and 15 are active in the

input and the other nibbles are inactive. Note that the right most nibble stands for the 15^{th} nibble and the leftmost nibble is the 0^{th} nibble in this representation. Similarly in the input and output representation, each digit shows the value for that nibble. As an example $[0000000000007604]$ means that the 15^{th} nibble has a difference of value 4. The 13^{th} nibble has a difference of value 6 and the 12^{th} nibble has a difference of value 7. From the above results, one conclude that the underlying block cipher makes LAC susceptible to forgery attacks where forged messages can be produced by utilizing the found differentials which hold with relatively high probability than what is claimed by the cipher designers.

Chapter 5

Conclusions and Future Work

Block ciphers, hash functions and authenticated encryption schemes are important symmetric key primitives that are used as building blocks in many security applications. This thesis focused on the cryptanalysis of some of these primitives including the Lesamnta-512 and DHA-256 hash functions, and the LAC authenticated encryption schemes.

In particular, we investigated the resistance of the Lesamnta-512 underlying block cipher against impossible differential attacks and studied the resistance of the DHA-256 compression function against collision attacks. Furthermore, we studied the resistance of LAC against forgery attacks that utilize high probability differentials in its underlying block cipher.

We can further continue the current work in the following directions:

1. In this work, we only considered the underlying block cipher of Lesamnta-512. For future research, the implications of the developed key recovery techniques can be extended to Lesamnta when used in the secret key set-up to construct different MAC

schemes.

2. One can study how to improve the results provided in this thesis, for example by extending the number of attacked rounds or improving the complexity of the presented attacks.

3. The impossible differential cryptanalysis applied to Lesamnta-512 can be performed on other hash functions to investigate the security margin of their underlying block ciphers.

4. The so-called automated techniques that were used throughout the thesis still require a lot of manual expert intervention in order to be applied to a specific block cipher or hash function. Further automation of these techniques is a very interesting and challenging research project that can have a large impact on the analysis and design of symmetric key systems.

5. The automated methods that were used throughout the thesis have the limitations that they cannot be applied to many ciphers because of their inherent computational complexity. Improving the computational efficiency of these automated cryptanalysis methods and developing new ones is another interesting research direction.

Bibliography

[1] lp-solve 5.5.2.0 Documentation, http://web.mit.edu/lpsolve/doc/.

[2] ISO/IEC 19772:2009. Information Technology - Security techniques Authenticated Encryption. 2009.

[3] M. Bellare, T. Kohno, and C. Namprempre. Authenticated encryption in SSH: provably fixing the SSH binary packet protocol. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 1–11. ACM, 2002.

[4] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology-ASIACRYPT 2000, volume 1976 of Lecture Notes in Computer Science*, pages 531–545. Springer-Verlag, 2000.

[5] E. Biham, A. Biryukov, and A. Shamir. Miss in the middle attacks on IDEA and Khufu. In *Fast Software Encryption (1999). Lecture Notes in Computer Science*, pages 124–138. Springer-Verlag, 1999.

[6] E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.

[7] A. Bogdanov, F. Mendel, F. Regazzoni, V. Rijmen, and E. Tischhauser. ALE: AES-based lightweight authenticated encryption. In *Fast Software Encryption (2013). Lecture Notes in Computer Science*. Springer-Verlag, 2013.

[8] C. Bouillaguet, O. Dunkelman, G. Leurent, and P. Fouque. G.: Another look at the complementation property. In *Fast Software Encryption (2010). Lecture Notes in Computer Science*. Springer-Verlag, 2010.

[9] J. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-damgård revisited: How to construct a hash function. In *CRYPTO (2005), Lecture Notes in Computer Science*, volume 3621, pages 430–448. Springer-Verlag, 2005.

[10] J. Daemen and V. Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer-Verlag, 2002.

[11] A. Biryukov E. Biham and and A. Shamir. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In *Advances in Cryptology-Eurocrypt'99, J. Stern, Ed., LNCS 1592,*, pages 12–23. Springer-Verlag, 1999.

[12] D. Gligoroski, R. Ødegard, M. Mihova, S. Knapskog, A. Drápal, V. Klima, J. Amundse, and M. El-Hadedy. Cryptographic hash function Edon-R'. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html, 2009.

[13] R. Grinberg. Bitcoin: an innovative alternative digital currency. *Hastings Science & Technology Law Journal*, 4:159, 2012.

[14] S. Hirose, H. Kuwakado, and H. Yoshida. SHA-3 proposal: Lesamnta. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html, 2008.

[15] S. Hirose, H. Kuwakado, and H. Yoshida. A minor change to Lesamnta-change of round constants-. www.hitachi.com/rd/yrl/crypto/lesamnta/, 2009.

[16] S. Hirose, H. Kuwakado, and H. Yoshida. Security analysis of the compression function of Lesamnta and its impact. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/documents/LESAMNTA_Comments.pdf, 2009.

[17] C. Jutla. Encryption modes with almost free message integrity. In *Advances in Cryptology-EUROCRYPT 2001, B. Pfitzmann (ed.), LNCS 2045*, pages 529–544. Springer-Verlag, 2001.

[18] J. Kim, S. Hong, and J. Lim. Impossible differential cryptanalysis using matrix method. *Discrete Mathematics*, 310(5):988–1002, 2010.

[19] H. Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is SSL?). In *Advances in Cryptology-CRYPTO'01, volume 2139 of Lecture Notes in Computer Science*, pages 310–331. Springer-Verlag, 2001.

[20] J. Lee, D. Chang, H. Kim, E. Lee, D. Hong, J. Sung, S. Hong, and S. Lee. A new 256-bit hash function DHA-256: Enhancing the security of SHA-256. In *Cryptographic Hash Workshop hosted by NIST*, 2005.

[21] G. Leurent. Differential Forgery Attack against LAC. https://hal.inria.fr/hal-01017048, July 2014.

[22] Y. Luo, X. Lai, Z. Wu, and G. Gong. A unified method for finding impossible differentials of block cipher structures. *Information Sciences*, 263:211–220, 2014.

[23] Martin M. gren, Martin Hell, Thomas Johansson, and Willi Meier. Grain-128a: a new version of Grain-128 with optional authentication. *International Journal of Wireless and Mobile Computing*, 5(1):48–59, 2011.

[24] Alfred J Menezes, Paul C V. Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[25] Ivica Nikoli and Alex Biryukov. Collisions for step-reduced SHA-256. In *Fast Software Encryption*, pages 1–15. Springer-Verlag, 2008.

[26] P. Rogaway. Authenticated-encryption with associated-data. In *Proceedings of the 9th CCS*, pages 98–107. ACM, 2002.

[27] Phillip Rogaway, Mihir Bellare, and John Black. OCB: A block-cipher mode of operation for efficient authenticated encryption. *ACM Transactions on Information and System Security (TISSEC)*, 6(3):365–403, 2003.

[28] X. Wang, Y. Yin, and H. Yu. Finding collisions in the full SHA-1. In *Advances in Cryptology–CRYPTO 2005*, pages 17–36. Springer-Verlag, 2005.

[29] X. Wang and H. Yu. How to break MD5 and other hash functions. In *Advances in Cryptology–EUROCRYPT 2005*, pages 19–35. Springer-Verlag, 2005.

[30] Wenling Wu, Lei Zhang, editors J. Lopez, and G. Tsudik. LBlock: a lightweight block cipher. In *Applied Cryptography and Network Security*, volume 6715, pages 327–344. Springer-Verlag, 2011.

[31] J. Zhong and X. Lai. Preimage attack on reduced DHA-256. *Journal of Information Science and Engineering*, 27(4):1315–1327, 2011.