IMPOSSIBLE AND IMPROBABLE DIFFERENTIAL CRYPTANALYSIS OF SPOOK ALGORITHM

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

BY

ONUR BOLEL

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

JUNE 2021

Approval of the thesis:

IMPOSSIBLE AND IMPROBABLE DIFFERENTIAL CRYPTANALYSIS OF SPOOK ALGORITHM

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

Prof. Dr. Deniz ZEYREK BOZŞAHİN
Dean, **Graduate School of Informatics** _____

Assist. Prof. Dr. Cihangir TEZCAN
Head of Department, **Cyber Security** _____

Assist. Prof. Dr. Cihangir TEZCAN
Supervisor, **Cyber Security Dept., METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Cengiz ACARTÜRK
Cognitive Science Dept., METU _____

Assist. Prof. Dr. Cihangir TEZCAN
Cyber Security Dept., METU _____

Prof. Dr. Ali Aydın SELÇUK
Computer Engineering Dept., TOBB ETÜ _____

Date: **14.06.2021**

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

Name, Last Name: Onur BOLEL

Signature : _____

ABSTRACT

IMPOSSIBLE AND IMPROBABLE DIFFERENTIAL CRYPTANALYSIS OF SPOOK ALGORITHM

BOLEL, Onur

MSc., Department of Cyber Security

Supervisor: Assist. Prof. Dr. Cihangir TEZCAN

June 2021, 78 pages

In recent years, the number of IoT devices increased considerably and the security of IoT devices became an important issue. Furthermore, most IoT devices have constrained resources in terms of memory, area and power. Therefore, cryptographic algorithms that provide their security should be suitable for the implementation on the constrained devices.

In 2013, NIST initiated a lightweight cryptography project to define the standards of lightweight cryptography. In 2018, the lightweight cryptography project turned into a competition-like process to choose the most convenient algorithms for constrained devices as a NIST standard. 57 algorithms were applied to the project. NIST published all algorithms for public evaluation and encouraged third-party analyses to reveal the weaknesses of algorithms. 32 algorithms were chosen as round 2 candidates.

In this thesis, we have investigated the Spook algorithm, which is one of the round 2 candidates of the NIST's lightweight cryptography competition. Spook is an AEAD algorithm that uses duplex sponge construction and tweakable block cipher. Besides, Spook has an internal permutation which is Shadow-512. We have worked on Shadow-512 permutation to find a distinguisher. Shadow-512 permutation was designed as 6-Step. The outputs of Shadow-512 permutation should seem random after the 6-Step operation. However, we have found two different 6-Step impossible differential distinguishers that cover full Shadow-512. Besides, we have found 7-Step impossible

distinguisher and 8-Step improbable distinguisher by adding one or more additional steps to Shadow-512. The 8-Step improbable differential covers the largest number of steps of Shadow-512 compared to previously found distinguishers in other published papers. To conclude, we can distinguish 6-, 7-, 8-Step of Shadow-512 from a random permutation by using our distinguishers.

Keywords: Lightweight Cryptography, differential cryptanalysis, impossible differential, improbable differential, Shadow-512

ÖZ

SPOOK ALGORİTMASININ İMKANSIZ VE OLASI OLMAYAN DİFERANSİYEL KRİPTANALİZİ

BOLEL, Onur

Yüksek Lisans, Siber Güvenlik Bölümü

Tez Yöneticisi: Dr. Öğretim Üyesi Cihangir TEZCAN

Haziran 2021, 78 sayfa

Son yıllarda, IoT cihazlarının sayısı oldukça arttı ve IoT cihazlarının güvenliği önemli bir konu haline geldi. Ayrıca, çoğu IoT cihazı, bellek, alan ve güç açısından kısıtlı kaynaklara sahiptir. Bu yüzden, güvenliklerini sağlayan kriptografik algoritmalar kısıtlı cihazlarda uygulanmaya elverişli olmalıdır.

2013 yılında NIST, hafif kriptografi standartlarını tanımlamak için bir hafif kriptografi projesi başlattı. 2018 yılında, hafif kriptografi projesi, kısıtlı cihazlar için en uygun algoritmaları NIST standardı olarak seçmek için yarışma benzeri bir sürece dönüştü. Projeye 57 algoritma başvurdu. NIST tüm algoritmaları herkesin değerlendirmesi için yayınladı ve algoritmaların zayıflıklarının ortaya çıkması için üçüncü taraf analizlerini teşvik etti. 32 algoritma 2. tur adayları olarak seçildi.

Bu tezde, NIST'in hafif kriptografi yarışmasının 2. tur adaylarından biri olan Spook algoritmasını inceledik. Spook, dubleks sünger yapısı ve ayarlanabilir blok şifre kullanan bir kimlik doğrulamalı şifreleme algoritmasıdır. Ayrıca, Spook'un Shadow-512 olan dahili bir permütasyonu vardır. Bir ayırt edici bulmak için Shadow-512 permütasyonu üzerinde çalıştık. Shadow-512 permütasyonu 6-Basamak olarak tasarlanmıştır. Shadow-512 permütasyonunun çıktıları 6-Basamak işlemden sonra rastgele olarak görünmelidir. Yine de, tam Shadow-512'yi kapsayan iki farklı 6-Basamak imkansız diferansiyel ayırt edici bulduk. Ayrıca, Shadow-512'ye bir ya da iki basamak ekleyerek 7-Basamak imkansız ayırt edici ve 8-Basamak olası olmayan ayırt edici bulduk. 8-Basamak olası olmayan diferansiyel ayırt edici, diğer yayınlanmış

makalelerdeki daha önceden bulunmuş ayırt edicilerle karşılaştırıldığında Shadow-512'nin en fazla basamağını kapsar. Sonuç olarak, ayırt edicilerimizi kullanarak 6-,7-,8-Basamak Shadow-512'yi rastgele permütasyondan ayırt edebiliriz.

Anahtar Sözcükler: Hafif Kriptografi, diferansiyel kriptanaliz, imkansız diferansiyel, olası olmayan diferansiyel, Shadow-512

To My Family

ACKNOWLEDGMENTS

First of all, I would like to express my sincere appreciation to my thesis supervisor Assist. Prof. Dr. Cihangir TEZCAN for boundless help, guidance, patience, suggestions, and feedback throughout this project.

I would like to thank Assoc. Prof. Dr. Cengiz ACARTÜRK and Prof. Dr. Ali Aydın SELÇUK for participating and contributing the thesis defense jury.

I am indebted to my family for always trusting and supporting me.

I would like to thank ASELSAN Inc. for supporting roles.

Last but not least, I also would like to thank my friend Onur ŞANAL for their extreme support and understanding.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

NIST	National Institute of Standards and Technology
AEAD	Authenticated Encryption with Associated Data
SPN	Substitution Permutation Network
XOR	Exclusive OR
TBC	Tweakable Block Cipher

CHAPTER 1

INTRODUCTION

Cryptology is a science that aims to protect information against third parties when two parties are communicating in an insecure channel. Cryptology can be divided into two subcategories: Cryptography and Cryptanalysis. Cryptography aims to design secure algorithms that can be used to encrypt information. On the other hand, Cryptanalysis aims to exploit the encrypted messages to reveal the information by using the weaknesses of the cryptographic algorithms.

To understand the whole picture, some terms are defined. The *plaintext* is the messages that are wanted to be protected against unauthorized parties and plaintext is generally denoted by P. If plaintext is encrypted with a cryptographic algorithm, a *ciphertext* is obtained. A ciphertext should be incomprehensible for everyone and it must look like a random sequence of characters. Also, one of the essential components of cryptography is k-bit *key*. The key is used to encrypt the plaintext to obtain ciphertext. The security of the cryptographic algorithm depends on the key, and the key must always be secret even if cryptography algorithms are publicly known, according to *Kerckhoff's Principle.* Initialization vector IV and cryptographic nonce N that should be random are the other inputs of a cryptographic algorithm. Nonce should be used only once in a communication session. In addition, t-bit tag T is used for authentication of messages. In a communication, the sender encrypts the plaintext and computes the tag T by using the key and then sends them to the receiver. The receiving end computes T' by using ciphertext then checks $T = T'$. If the tags match, it means messages are not altered by third parties.

Cryptography includes several subjects such as Symmetric Cryptography, Asymmetric Cryptography and Hash Functions. Symmetric and Asymmetric Cryptography differ by the usage of a key of cryptographic algorithms. Asymmetric Cryptography algorithms use two different keys, which are called private and public keys. The plaintext is encrypted with one of the keys and the ciphertext should be decrypted with the other key. As the name indicates, the public key is publicly known and the private key is only obtained by the owner. Besides, asymmetric cryptography algorithms are also used for authentication and key exchange. RSA (Rivest et al., 1978), Diffie-Hellman (Diffie & Hellman, 1976) and Elliptic Curve

Cryptography (Araki et al., 1998) are the most known asymmetric cryptography algorithms. On the other hand, in symmetric cryptography, two parties have the same key and both encryption and decryption processes are operated with the same key. Block ciphers and stream ciphers are two main categories of Symmetric Cryptography.

In this chapter, firstly, block ciphers and their types will be explained. Then some cryptanalysis techniques that are used to exploit the weaknesses of block ciphers will be clarified. After that, the concept of lightweight cryptography and NIST's lightweight cryptography competition will be mentioned.

1.1. Block Ciphers

A block cipher algorithm takes b-bit input and produces b-bit output. To do that, the plaintext is divided into b-bit blocks, and each block is encrypted with k-bit key. The ciphertext is composed of these encrypted blocks. Block cipher algorithms have a round function that iterates the input for r times. In other words, the round function is applied to b-bit blocks for r rounds and then ciphertext blocks are obtained. Also, round keys are generated from the key to use in each round. AES (Daemen & Rijmen, 2002), DES, PRESENT (Bogdanov et al., 2007) are the most known block cipher algorithms.

Each block cipher algorithm has a different design and round function; however, it is possible to group the designs into two categories which are Feistel Networks and Substitution Permutation Networks (SPN). In addition, some algorithms use Sponge Construction as a block cipher.

SPN consists of three main components, which are key addition, substitution and permutation. Firstly, the plaintext block is XORed with the round key. Then, substitution is applied. The substitution layer consists of $a \times b$-bit S-Boxes, which provide confusion. After that, the permutation layer which provides diffusion is applied. These three layers are applied for each round. After r rounds, the output is XORed with the last round key to produce the ciphertext block. AES (Daemen & Rijmen, 2002), PRESENT (Bogdanov et al., 2007) and SERPENT (Biham, Anderson, et al., 1998) are SPN type block ciphers. In this thesis, the Shadow-512 permutation of the Spook (Bellizia et al., 2019) algorithm was investigated. Shadow-512 has an SPN structure. Fig. 1.1 shows the general SPN structure.

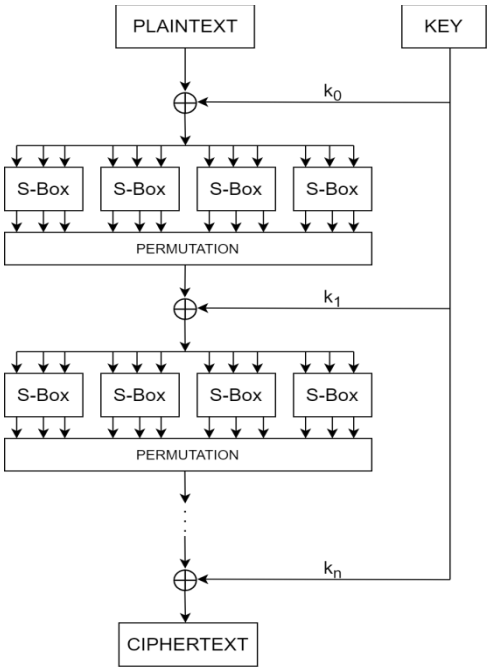

Figure 1.1: Substitution Permutation Network

Feistel Networks has two parts which are round function and swap operation. Firstly, the plaintext block is divided into two pieces. One of the pieces and round key are the input of the round function. Round function is applied to one of the pieces and the output of the round function is XORed with the other piece. Then two pieces are swapped with each other. HIGHT (D. Hong et al., 2006) has a Feistel Network structure. Fig. 1.2 shows the Feistel Network.

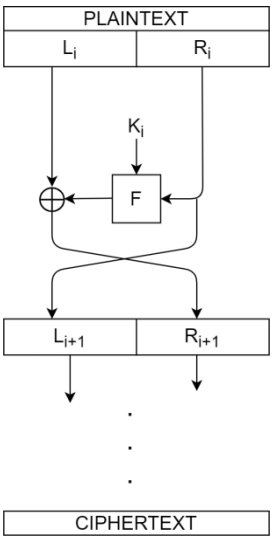

Figure 1.2: Feistel Network

Sponge function was first introduced in (Bertoni et al., 2007) as a hash function. Then, it was used in the design of Keccak family (Bertoni et al., 2009) that was chosen as the algorithm of the SHA-3 hash function. After that, Duplex Sponge construction was introduced in (Bertoni et al., 2012) as an authenticated encryption

mode. It can be said that the design is similar to block cipher. However, the key schedule part does not exist. The duplex sponge construction relies on a strong permutation. In duplex construction, the $r + c$-bit state has two parts such as r-bit rate and c-bit capacity. Firstly, $r + c$-bit state initializes to zero. The initialization part may be different for different algorithms. Then input block P_0 is padded to r-bit. After that, P_0 is XORed with r-bit of state. The permutation f is applied to the state. The first t-bit of the state gives output. Fig. 1.3 shows the duplex sponge construction.

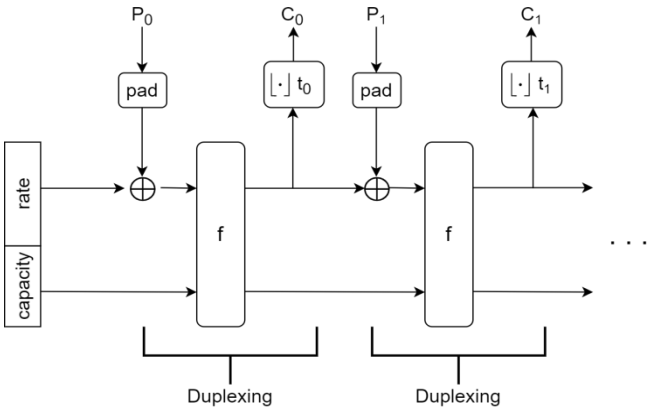

Figure 1.3: Duplex Construction

In this thesis, we investigated the Spook (Bellizia et al., 2019) algorithm and it has duplex sponge construction. In addition, the permutation f corresponds to Shadow-512 permutation. A detailed explanation about the Spook algorithm will be given in Chapter 2.

1.2. Cryptanalysis of Block Cipher

According to *Kerckhoff's Principle,* the key must always be kept secret in a cryptosystem even if cryptography algorithms can be known publicly. Therefore, most of the time, an attacker tries to find the correct key that is used in the encryption algorithm. There are several generic attacks that are performed to obtain key material. Exhaustive search is one of the most known attack types in cryptanalysis. An attacker captures a plaintext-ciphertext pair and tries to encrypt plaintext with all possible keys until correct ciphertext is observed. Similarly, if only a ciphertext is captured, the ciphertext can be decrypted with all possible keys until meaningful plaintext is observed. If the key has k-bit, there are 2^k possible keys. Therefore, the required time in the worst-case scenario is directly proportional to the key length. After all, the exhaustive search may not be practical if the key length is large.

All plaintext blocks and corresponding ciphertext blocks for a key can be captured and stored in a large memory. If a ciphertext wanted to be decrypted, the corresponding plaintext is obtained by searching in the memory. These types of attacks are called table attacks. For a block cipher, both plaintext and ciphertext blocks are b-bit blocks. There are 2^b possible plaintext-ciphertext pairs. If b is large, it is hard to find the memory space to store all data.

There is another technique that aims to use less memory and time. Hellman introduced Time Memory Trade-off Attack (Hellman, 1980) and it aims to reduce the required time by using pre-computed data. In this attack, only a small part of the data is pre-computed and stored in the memory. Therefore, when an attacker performs an exhaustive search to a ciphertext, an attacker needs less than 2^k operations to obtain the plaintext that exists in the table.

The logic behind cryptanalysis is finding the correct key or revealing the secret information from encrypted data. As mentioned above, an exhaustive search can take more time than an attacker has. Also, a table attack can need a huge amount of memory. If an attack that reveals the correct key of a cipher needs less time than exhaustive search and less memory than a table attack, it can be said that the cipher is *broken.*

The attack types can be categorized according to the data that is used. *Ciphertext-only* or *known-ciphertext* attacks mean that an attacker has only some ciphertexts to find the correct key or corresponding plaintext. Exhaustive search is an example of *ciphertext-only* attacks. In *known-plaintext* attacks, an attacker has both plaintext and ciphertext pairs. Table attack is an example of *known-plaintext attack.* In *chosen-plaintext* attacks, an attacker chooses some plaintexts to be encrypted, then captures the ciphertexts and makes some calculations on them to find the key. *Adaptive chosen-plaintext* attacks look like *chosen-plaintext* attacks. Firstly, attackers choose some plaintexts and obtain their corresponding ciphertexts. After performing an analysis of ciphertexts, attackers choose new plaintexts to encrypt and they improve their attacks by analyzing the new ciphertexts.

It is possible to compare attacks in terms of resources that are needed. The volume of data that is needed to perform an attack is defined as *data complexity.* The number of plaintexts or ciphertexts that are used in an attack gives information about *data complexity.* Some attacks need more operation in software or hardware. *Time complexity* describes the computational time that it takes to perform an attack. To perform some attacks, the data should be stored in a memory. The volume of storage gives the *memory complexity* of an attack.

1.3. Differential Cryptanalysis

Differential cryptanalysis was first introduced in (Biham & Shamir, 1991) in the 1980s. It is a statistical chosen-plaintext attack that analyzes the relation between input and output differences. It aims to find a pattern which input differences lead to which output differences by using the same key. The difference is obtained by XORing two messages. Let P and P' be two inputs of an algorithm and C and C' be two outputs after r rounds, respectively. $P \oplus P' = \alpha$ denotes the input difference and $C \oplus C' = \beta$ denotes the output difference. β has k bits and the probability of having β output difference is $p = 2^{-k}$ for a random permutation. If an α input difference leads β output difference after r rounds with the probability $p_0 > p$, it is considered as differential characteristic of an algorithm for r rounds. Also, this statistical property is called a *distinguisher* which can be used to distinguish the r rounds of a cipher from a random permutation. The output of an algorithm should be

random and cannot be predicted by an adversary. A distinguisher helps the adversary notice whether the output is random or the algorithm's output itself.

Differential distinguisher, which is mentioned above, can be used for key-guessing. Firstly, a distinguisher is found for r rounds of the algorithm by an adversary. Then, one or more rounds of encryption are added to the top or bottom of the distinguisher. An adversary collects or generates N input pairs and their corresponding output pairs. Then, the adversary checks N input-output pairs to determine how many times the distinguisher is obtained for candidate keys. The number of occurrences of the distinguisher is counted for every possible key. As mentioned above, p_0 denotes the probability of obtaining the distinguisher for a correct key and p denotes the probability of obtaining the distinguisher for a random permutation. In addition, wrong keys can be considered behaving like a random permutation. This approach is called Wrong-key Randomization Hypothesis. After all, an adversary has two binomial distributions with parameters (N, p_0) for a correct key and (N, p) for a wrong key. The expected values of two binomial distributions are $E = N \cdot p$ and $E_0 = N \cdot p_0$. Then, the threshold T should be specified between E and E_0 to determine whether the key is correct or wrong. The key counter should be bigger than T for a correct key and smaller than T for a wrong key. However, the key counter may be smaller than T for some possible correct key. It is called non-detection and the probability of non-detection is denoted by p_{nd}. Likewise, the key counter may be bigger than T for some possible wrong keys. It is called false alarm and the probability of false alarm is denoted by p_{fa}. Hence, the success probability of an attack is $1 - p_{nd}$. Since the false alarm gives the wrong information about whether the key is correct or wrong, it causes to make extra effort to find the correct key. If p_{fa} increases, data complexity increases. Therefore, an adversary wants to set p_{fa} and p_{nd} to a very small value. To do this, an adversary should choose N big enough. If N increases, the difference between E and E_0 increases. Therefore, p_{fa} and p_{nd} get smaller. However, since N is the number of the input pairs, if N increases, the data complexity of the attack increases. Therefore, the adversary should determine the optimal N value. The optimal N value can be determined by setting p_{fa} and p_{nd} very close to zero. The false-alarm and non-detection probabilities can be found by solving the equations (1.1) and (1.2).

$$p_{nd} = \sum_{k=0}^{T} \binom{N}{k} p_0^k (1 - p_0)^{(N-k)} \qquad (1.1)$$

$$p_{fa} = \sum_{k=0}^{T} \binom{N}{k} p^k (1 - p)^{(N-k)} \qquad (1.2)$$

1.4. Truncated Differential Cryptanalysis

Truncated differential cryptanalysis was introduced by (Knudsen, 1994). It can be considered as a special type of differential cryptanalysis. The differential analysis

aims to guess n-bit output difference of n -bit ciphertext pairs. On the other hand, in truncated differentials, all input and output bit differences are not explicitly defined. The aim is to find only part of the difference of outputs. Even obtaining 1-bit output difference is sufficient.

1.5. Impossible Differential Cryptanalysis

Impossible differential cryptanalysis was first introduced in Crypto '98 Rump Sessions by (Biham, Biryukov, et al., 1998). Thereafter, Biham, Biryukov and Shamir performed an impossible differential attack (Biham et al., 1999) on reduced round Skipjack (NIST, 1998). There are many examples of impossible differential attacks on block ciphers. In (Tsunoo et al., 2008), they found 9-round impossible differentials of CLEFIA (Shirai et al., 2007) which was developed by Sony Corporation. In (Tezcan, 2016), 5-round impossible differential was found on ASCON (Dobraunig, Mendel, et al., 2019), which is one of the finalists of the NIST's Lightweight Cryptography Competition.

Impossible differential cryptanalysis aims that the r round truncated differential cannot be obtained at the output. The probability p_0 which was mentioned in Section 1.3 should be zero for an impossible differential attack. To obtain impossible differential distinguisher, the miss-in-the-middle technique is used. Firstly, α input difference is given to input pairs and then α input difference leads to β difference after r_1 rounds in the forward direction with probability one. Similarly, δ output difference leads γ difference after r_2 rounds in the backward direction with probability one. If β and γ do not match in the middle, it can be said that α input difference does not become δ output difference after $r_1 + r_2$ in the forward direction with probability one. Fig. 1.4 shows the miss-in-the-middle approach.

$$\alpha \xrightarrow{p_1=1} \beta \neq \gamma \xleftarrow{p_2=1} \delta$$

Figure 1.4: Miss-in-the-middle technique

The cipher can be distinguished from a random permutation thanks to the impossible differentials. If it is observed that the α input difference leads δ output difference after $r_1 + r_2$ rounds, it can be said that the output does not belong $r_1 + r_2$ round version of the cipher. For r round impossible differential distinguisher where $\alpha \nrightarrow \delta$, assume that the output difference δ has k many fixed bits. The probability of not obtaining δ output difference is $1 - 2^{-k}$ for one pair of random permutation output. If we use $c \cdot 2^k$ pairs, the probability becomes $(1 - 2^{-k})^{c \cdot 2^k}$. To find the approximate value of this probability, equation (1.3) is used.

$$(1 - \frac{1}{2^k})^{c \cdot 2^k} \approx \left(\frac{1}{e}\right)^c = \frac{1}{e^c} \qquad (1.3)$$

The approximation can be obtained by using the equation (1.4).

$$e^x = \lim_{n\to\infty}(1+\frac{x}{n})^n \qquad (1.4)$$

In this thesis, we found 6-Step and 7-Step distinguisher of Shadow-512 permutation by using impossible differential cryptanalysis. The detailed explanations will be given in Sections 4.3 and 4.4 about these distinguishers.

1.6. Improbable Differential Cryptanalysis

Improbable differential cryptanalysis was first introduced in (Tezcan, 2010). Tezcan performed 13,14,15-round improbable differential attacks on CLEFIA (Shirai et al., 2007), which has 128-bit,192-bit and 256-bit key lengths, respectively. In (Tezcan, 2014), Tezcan performed a 13-round improbable differential attack on PRESENT (Bogdanov et al., 2007), which is included as an international standard in (*ISO/IEC 29192-2:2019*, 2019).

Some differentials are less observable than a random permutation at the output. These kinds of differentials establish the idea of improbable differentials. Typically, an attacker desires to obtain δ output difference with high probability than random permutation when the input difference is α for differential attacks. For improbable differential attacks, the purpose is to find any output difference other than δ if the input difference is α. Let us denote the probability of observing non δ output difference as p'. The probabilities of observing $\alpha \rightarrow \delta$ transition are denoted as p and p_0 for a random permutation and the cipher, respectively. For improbable differential, an attacker expects that $p_0 < p$ since δ output difference is less likely to exist for the cipher than random permutation. To conclude, the probability of observing δ output difference if input difference is α can be defined according to equation (1.5).

$$p_0 = p \cdot (1 - p') \qquad (1.5)$$

An attacker should ensure that α input difference leads δ output difference with a low probability than random permutation. On the contrary case, p_0 becomes bigger than $p \cdot (1 - p')$. Besides, p' denotes the probability of observing any output difference other than δ, it is expected that $p' = 1$ for an impossible differential because α input difference never leads δ output difference. It makes $p_0 = 0$, which means $\alpha \rightarrow \delta$ is impossible. Therefore, it can be said that the impossible differential attacks are a special case of improbable differential attacks.

In (Tezcan, 2010), Tezcan introduced *almost miss in the middle* technique. It looks like a miss-in-the-middle technique; however, the probabilities p_1 and p_2 are different than 1. After r_1 round in the forward direction, α input difference leads β difference with the probability p_1 and after r_2 rounds in the backward direction δ output difference leads γ difference with probability p_2. If β and γ do not match in the middle, it can be said that $\alpha \rightarrow \delta$ transition cannot be obtained with the probability $p' = p_1 \cdot p_2$ after $r_1 + r_2$ rounds. This approach can be used to cover more rounds than impossible differentials. If one or more differentials are added to top and bottom

of impossible differential, the improbable differential can cover more rounds with probability $p' = p_1 \cdot p_2$ as in the *almost miss in the middle* technique. This is called expansion method. Although the expansion method helps an attacker find longer differentials to attack more rounds, it increases the data and time complexities. Since p_0 increases, data complexity increases. Therefore, more time is needed to perform the attack. Fig. 1.5 shows the expansion method.

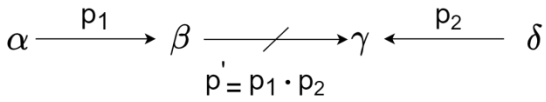

Figure 1.5: Expanded Improbable Differentials

1.7. Lightweight Cryptography Competition

Internet of Things (IoT) is a concept that includes sensor networks, embedded systems and software that are connected and communicated over the Internet. In recent years, IoT devices are widely used in areas such as smart homes, medical and health care systems, agriculture, monitoring systems, transportation, etc. Some of these devices have a strong capability to perform larger computations on them; however, some of IoT devices are highly-constrained in terms of memory, power consumption and performance. Sensors, RFID tags, smart cards, key fobs, electronic toll collection systems, biometric systems are the major examples that use highly-constrained devices. For example, some smart cards work with the absence of an internal power supply like a battery and they use electromagnetic fields to be powered on when they are attached to the RFID reader. Since the number of these types of constrained devices increases day by day, their security issues have become one of the crucial topics. Their data should be encrypted to provide confidentiality, integrity and availability. Cryptographic algorithms can be used to satisfy the security concern about them. However, many constrained devices use low-power MCUs which also have a small size of RAM. Implementation of a cryptographic algorithm on them can be considered as an overhead for their actual purposes. In fact, it can be said that a cryptographic algorithm such as AES (Daemen & Rijmen, 2002) that uses many logic gates and needs larger RAM is not suitable on these constrained devices. Therefore, algorithms that have smaller permutations, smaller block and key sizes, simpler key schedules are more favorable for constrained devices. Such algorithms are called lightweight cryptographic algorithms. Most lightweight designs satisfy different implementation constraints and serve different purposes. PRESENT (Bogdanov et al., 2007), HIGHT (D. Hong et al., 2006) and CLEFIA (Shirai et al., 2007) aim low hardware footprint in their lightweight designs. On the one hand, PRIDE (Albrecht et al., 2014) and SPECK (Beaulieu et al., 2015) aim low memory usage on embedded processors. In (Alex Biryukov & Perrin, 2017), they performed a survey about lightweight designs to systematize the concept of lightweight algorithms by investigating more than 100 designs.

The most comprehensive and worldwide project about lightweight cryptography was initiated by the National Institute of Standards and Technology (NIST) in 2013 to standardize lightweight cryptographic algorithms for constrained devices. In 2015,

First Lightweight Cryptography Workshop was held by NIST to define the requirements of lightweight cryptography. In 2016, NIST held the second Lightweight Cryptography Workshop, and then in 2017, NIST published a report on Lightweight Cryptography (McKay et al., 2017). Since current NIST's cryptographic standards do not meet some requirements, which are performance, side-channel resistance, hardware and software specific metrics for constrained devices, they decided to start a competition for lightweight cryptography. In 2018, they announced submission requirements for candidate algorithms. The minimum acceptability requirements that were defined in (NIST, 2018) are as follows:

- **AEAD requirements:** NIST expected that the candidate algorithms had authenticated encryption with associated data scheme (AEAD). The minimum size of key, nonce and tag should be 128-bit, 96-bit and 64-bit, respectively. The candidate AEAD algorithms should provide confidentiality against forgery attacks if the nonce is used only once. Besides, plaintext should not be produced in the decryption process if the tag is invalid.

- **Hash function requirements:** It is optional that candidate algorithms can have hashing functionalities. The minimum output length should be 256-bit. The hash function should be resistant to collision and second preimage attacks. At least 2^{112} computations can be performed for attacks on the hash function.

- **Design requirements:** The AEAD algorithm and optional hash function should be suitable for constrained devices. Algorithms should be designed by considering the performance on 8-bit, 16-bit, 32-bit microcontrollers, FPGAs and ASICs. The algorithms which are resistant against side-channel attacks, timing attacks, simple and differential power analysis are desired.

Side-channel resistance, fault attack resistance, cost metrics (area, memory), performance measurements (latency, throughput, power consumption), suitability for both hardware and software are evaluation criteria that NIST defined. NIST also stated that third-party analysis of algorithms is desirable. After all, in 2019, 57 different algorithms were submitted to the competition and NIST approved 56 of them as first round candidates. The designs of candidate algorithms were shared with the public for third-party analysis.

In October 2019, NIST published the status report on first round candidates (Turan et al., 2019) and 32 of candidate algorithms were chosen as round 2 candidates. NIST gathered the criteria of selection in two main topics, which are maturity of the candidates and cryptanalysis of the candidates. Some algorithms were eliminated from the competition since their algorithms had no third-party analysis. NIST also stated that adequate analyses were not made to their designs by themselves to satisfy their security claims. Therefore, these algorithms were considered immature to standardize. On the one hand, some of the algorithms were eliminated from the competition due to third-party analyses. Significant weaknesses were found in their designs. The round 2 candidates are shown in Table 1.1.

Table 1.1: Round 2 Candidates of NIST's Lightweight Cryptography Competition

ACE	ASCON	COMET	DryGASCON
Elephant	ESTATE	ForkAE	GIFT-COFB
Gimli	Grain-128AEAD	HyENA	ISAP
KNOT	LOTUS-AEAD	mixFeed	ORANGE
Oribatida	PHOTON-Beetle	Pyjamask	Romulus
SAEAES	Saturnin	SKINNY-AEAD	SPARKLE
SPIX	SpoC	Spook	Subterranean 2.0
SUNDAE-GIFT	TinyJambu	WAGE	Xoodyak

After the analysis and performance evaluation of round 2 candidates, in March 2021, NIST announced the finalists of the lightweight standardization process. ASCON (Dobraunig, Mendel, et al., 2019), Elephant (Beyne et al., 2019), GIFT-COFB (Banik et al., 2019), Grain128-AEAD (Hell et al., 2019), ISAP (Dobraunig, Eichlseder, et al., 2019), Photon-Beetle (Bao et al., 2019), Romulus (Iwata et al., 2019), SPARKLE (Beierle et al., 2019), TinyJambu (Huang, 2019), and Xoodyak (Daemen et al., 2019) are the finalists of the competition. ASCON (Dobraunig, Mendel, et al., 2019) was also the first choice of the lightweight applications category of the CAESAR competition (*Cryptographic competitions: CAESAR submissions*, 2014). ASCON (Dobraunig, Mendel, et al., 2019), Elephant (Beyne et al., 2019), ISAP (Dobraunig, Eichlseder, et al., 2019), Photon-Beetle (Bao et al., 2019), SPARKLE (Beierle et al., 2019), Xoodyak (Daemen et al., 2019) are the finalists that have permutation based algorithm. GIFT-COFB (Banik et al., 2019) and TinyJambu (Huang, 2019) have block cipher based designs. Romulus (Iwata et al., 2019) is based on a tweakable block cipher design. Among finalist algorithms, the only algorithm that is based on stream cipher structure is Grain128-AEAD (Hell et al., 2019). It is expected that the final round will end up in a year.

1.8. Our Contribution and the Structure of the Thesis

In this thesis, we have investigated the Spook (Bellizia et al., 2019) algorithm, which is one of the round 2 candidates of NIST's Lightweight Cryptography competition. Spook is an AEAD algorithm that is based on a tweakable block cipher and duplex sponge construction. Also, Spook uses Shadow-512 as an internal permutation. In this thesis, we have worked on Shadow-512 permutation to find a distinguisher.

In Chapter 2, the design specifications of Spook and Shadow-512 permutation will be explained. In Chapter 3, the 5-Step truncated differential distinguisher on

Shadow-512 with probability one that was first introduced in (Derbez et al., 2020) will be explained. Although Shadow-512 permutation was designed as 6-Step, they found a distinguisher that covers more than six steps which can be considered as a round-extended variant of Shadow-512. They found 6-, 7-Step truncated differential distinguishers with probability $2^{-16.245}$. In Chapter 4, we tried finding impossible differential distinguishers of Shadow-512. We have found 6-, 7-Step impossible differential distinguishers by using the 5-Step truncated differential of (Derbez et al., 2020). Also, we have introduced the 8-Step improbable differential distinguisher. This is the first distinguisher that is found on Shadow-512 that covers 8-Step if Shadow-512 is considered as 8-Step.

CHAPTER 2

OVERVIEW OF SPOOK

In 2015, NIST initiated a Lightweight Cryptography project to standardize lightweight cryptography algorithms for constrained devices. Spook (Bellizia et al., 2019) is one of the Round-2 candidates of the competition among 32 different algorithms. The main purpose of Spook is to provide a secure design in terms of both low-cost implementation and side-channel analysis. Spook was designed as duplex sponge-based (Bertoni et al., 2012) authenticated encryption with associated data algorithm that operates in S1P mode of operation (C. Guo et al., 2019) to provide leakage resistance against side-channel attacks. The S1P mode of operation, which is Sponge One Pass, provides that data is processed only once to produce both ciphertext and tag in the encryption process. It is a significant advantage for a lightweight design. Moreover, the secret key is only used twice for both encryption and decryption processes in the S1P mode of operation. In (Daemen et al., 2017), they stated that the duplex sponge construction is beneficial to implement authenticated encryption algorithms in terms of providing leakage resistance. Also, the S1P mode of operation uses Tweakable Block Cipher Clyde-128 and Shadow-512 permutation that are both based on LS-design (Grosso et al., 2015), which consists of L-boxes and bitslice S-boxes. However, some improvements are made to previous LS-designs by using word-level L-Boxes instead of table look-up L-boxes. The LS-design of Clyde-128 and Shadow-512 has efficient slicing and masking to provide security against side-channel attacks, as mentioned in (Goudarzi & Rivain, 2017; Gross et al., 2017). Moreover, the Tweakable Block Cipher (TBC) guarantees data integrity in case of data leakage. TBC also provides multi-user security with the usage of randomly chosen public tweak.

In this chapter, firstly S1P mode of operation will be explained. Then Clyde-128 Tweakable LS-design and Shadow-512 permutation will be introduced. The components of Clyde-128 and Shadow-512 will be described in detail.

2.1. S1P mode of operation

S1P mode of operation, “Sponge One Pass,” is a leakage-resistant lightweight design that was introduced in (Berti et al., 2019) for AEAD schemes. S1P provides security

in case of nonce misuse and also makes the algorithm more resistant against side-channel leakage.

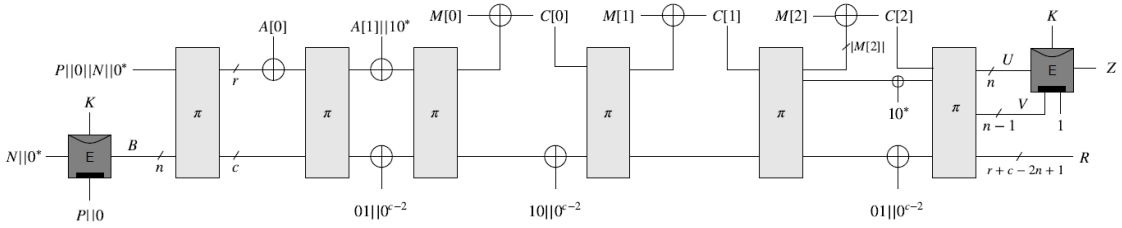

Figure 2.1: S1P mode with TBC E

Note. S1P mode with TBC E. Reprinted from "Spook: Sponge-based leakage-resistant authenticated encryption with a masked tweakable block cipher," by Bellizia et al., Lightweight Cryptography Standardization Process round 2 submissions, NIST. Reprinted with permission.

Fig. 2.1 shows the S1P mode of operation of Spook. Spook has a duplex sponge construction, so r and c represent the rate and capacity bits, respectively. The plaintext is denoted as M and M *is* divided into l blocks whose length is equal to r-bit except for the last block. The length of the last block of plaintext is between 1-bit and r-bit. A denotes the associated data and likewise the plaintext, associated data divided into r bits of blocks. The nonce is denoted as N, which is τ bits. Randomly chosen secret key K is n bits and public tweak P is $n-1$ bits. The last bit of the public tweak specifies whether single-user or multi-user. If the last bit of the public tweak P is set to 1 and the other bits of P is chosen randomly, the multi-user security is selected. On the other hand, the public tweak is set to zero for single-user security. For a multi-user version, public tweak is randomly chosen. The Tweakable Block Cipher, which is denoted by E, processes n-bit blocks. It takes $N \parallel 0^*$, $P \parallel 0$, K to produce n-bit initial seed B. π denotes the permutation that takes $(r+c)$-bit input. The primary parameters of Spook are $n = 128$, $r = 256$, $c = 256$, $\tau = 128$. Although each data that is processed in the S1P operation is considered as bitstring, Spook takes the input as bytestring. Therefore, firstly, bytestrings of input data are transformed to bitstrings. After the encryption or decryption process, the bitstrings of output data are transformed to bytestring to produce ciphertext or plaintext, respectively.

Encryption starts with a TBC call which is E. It takes $N \parallel 0^*$, $P \parallel 0$ and K as inputs and produces n-bit B as an initial seed. B can be considered as a key for duplex sponge construction to produce ciphertext blocks. Detailed explanations about TBC will be given in the following section. Concatenation of $P \parallel 0$ and $N \parallel 0^*$ which is $P \parallel 0 \parallel N \parallel 0^*$ can be considered as the initialization vector (IV) of the algorithm. The permutation state is composed of IV and B, which is $IV \parallel B$. After the first permutation part, r-bit associated data blocks except for the last block are processed with the rate bits of the state if associated data exists. The last block of associated data is padded with 10^* before the processing with the state. After processing the associated data, the state is XORed with $0^r \parallel 10 \parallel 0^{c-2}$ and then permutation is applied to the state. Now, the encryption procedure starts. The first r -bit of the state

is XORed with the first plaintext block to produce the first ciphertext block. Then, capacity bits of the state are concatenated with the ciphertext to obtain 512-bit state. The permutation is applied to the state and the other plaintext blocks are processed in the same way to produce ciphertext blocks. After that, there is another TBC call at the end of the operation to produce the tag Z. The inputs of the second TBC call are $U, V||1$ and secret key K. U is n-bit and V is (n-1)-bit. $U||V$ is the first $2n-1$bit of the permutation state. The purpose of the concatenation of 1 with V is to ensure that the tweak is different from the first TBC call. At the end of the procedure, the ciphertext, which is denoted by C is obtained by concatenation of ciphertext blocks and the tag.

Decryption starts with a TBC call to produce n-bit B as an initial seed, likewise in the encryption. At the end of the decryption operation, there is an inverse TBC call which takes the tag Z as an input to obtain U^*. If $U^* = U$, the operation starts to generate the plaintext. It can be said that the secret key K is only used twice for the encryption or decryption process. Moreover, there is not an extra effort to generate the tag while data is processing. Therefore, it can be said that the S1P mode of operation is single-pass.

2.2. Clyde-128, Tweakable LS-Design

As mentioned in the previous section, the S1P mode of operation uses a Tweakable Block Cipher to generate the authentication tag. The Tweakable LS-design of Clyde-128 is a part of SCREAM (Grosso et al., 2014) authenticated encryption with associated data algorithm. The TLS-design is a tweakable variant of LS-designs (Grosso et al., 2015) that use efficient masking and bitslice S-Boxes to mitigate the risks of side-channel attacks. Such LS-Designs work on x-bit state, which is denoted by $x = s \cdot l$. Here s denotes the size of the S-Box and l denotes the number of the columns. L-Box which is linear layer of Clyde-128 is composed of two rows and its size is $2l$. The state consists of s rows and l columns. For Clyde-128, the design is defined as $s = 4$ and $l = 32$ and therefore, the state is 128-bit. The step number of Clyde-128 is $N_s = 6$ and each step consists of two rounds. Each round starts with an S-Box operation, then L-Box is applied. After that, round constant is added to state. Besides, the Tweakable Block Cipher algorithm of Clyde-128 uses a tweakey. Tweakey framework introduces a new parameter tweak T that provides larger key space to a block cipher. Tweakey (Jean et al., 2014) is generated from the master key K and tweak T. The tweakable block cipher which was first introduced in (Liskov et al., 2011) takes the plaintext, key and tweak as input to produce ciphertext. The tweak T can be public. The tweak addition also provides resistance against to related-key attacks. Clyde-128 uses the SCREAM's (Grosso et al., 2014) tweakey scheduling algorithm that takes the 128-bit key and 128-bit tweak. Firstly, the tweak is parsed into two 64-bit pieces such that $T = t_0 \parallel t_1$ and then three different tweakeys are generated according to formulas (2.1), (2.2) and (2.3). Tweakey is denoted by TK.

$$TK(3r) = K \oplus (t_0 \parallel t_1) \quad (2.1)$$

$$TK(3r+1) = K \oplus (t_0 \oplus t_1 \parallel t_0) \quad (2.2)$$

$$TK(3r+2) = K \oplus (t_1 \parallel t_0 \oplus t_1) \quad (2.3)$$

In the S1P mode of operation, Clyde-128 takes $N \parallel 0^*$, $P \parallel 0$ and K as inputs. According to reference C code of Spook, $N \parallel 0^*$ is called the *padded nonce*. K is used as a secret key and $P \parallel 0$ is used as a tweak. In each step, a different tweakey is used. Firstly, *padded nonce* and the first tweakey are XORed to construct the state S. Then, round function is applied to the state. The second tweakey addition is applied at the end of the round function. This process is applied in each step. The components of the round function of Clyde-128 will be explained in the following sections.

2.3. Shadow-512

Shadow-512 can be considered as the permutation layer of the S1P mode of operation of the Spook algorithm. Shadow-512 also uses a variant of LS-designs, which is called multiple LS-designs. Shadow-512 works on $x = m \cdot s \cdot l$-bit state. Here m denotes the number of LS-designs which we call a *bundle*. s denotes the size of the S-box and l denotes the number of columns. For Shadow-512, $m = 4$, $s = 4$ and $l = 32$. The size of the state is 512-bit. In other words, there are four 128-bit bundles B_i $(0 \leq i < m)$ in the state. Shadow-512 permutation has a Substitution Permutation Network (SPN). The 512-bit state is updated by iterating six steps. Each step consists of two rounds which are Round A and Round B. Round A consists of S-Box, L-Box and round constant addition parts. Round B consists of S-Box, D-Box and round constant addition parts. There is also another variant of Shadow, which is Shadow-384. It has 384-bit state. The only difference between Shadow-512 and Shadow-384 is in the D-Box layer since Shadow-384 consists of three 128-bit bundles. In Fig. 2.2, the Shadow-512 permutation is shown.

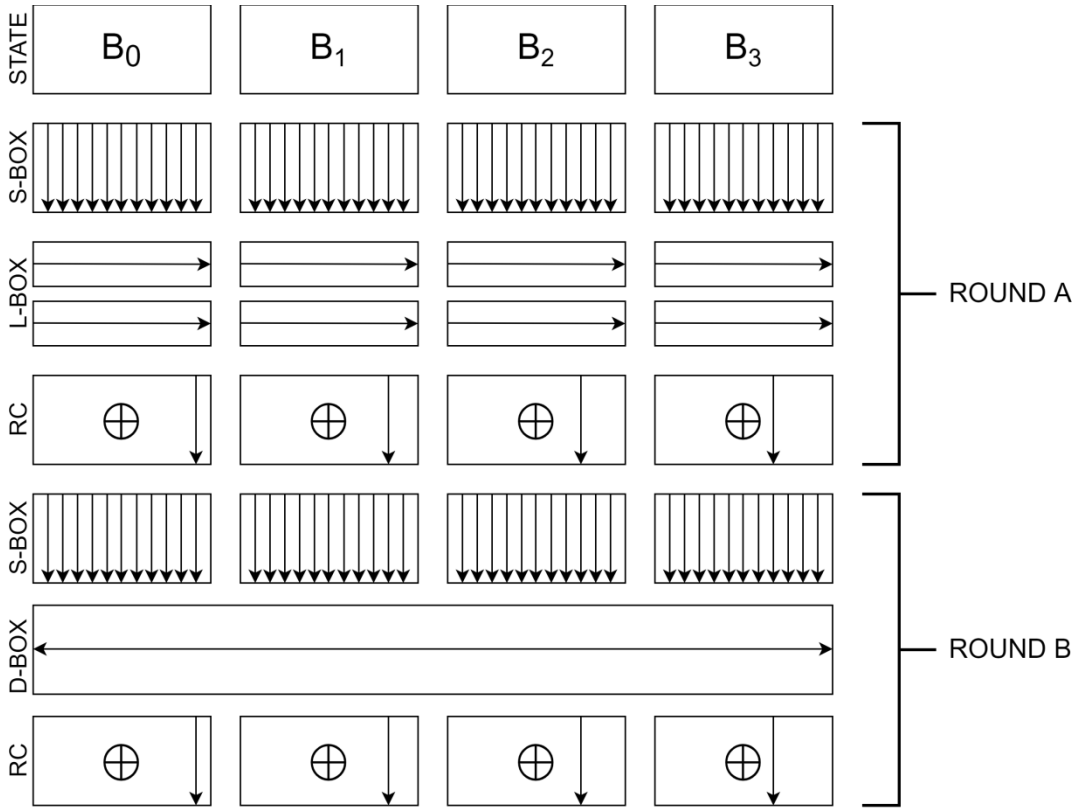

Figure 2.2: Illustration of Shadow-512

In Fig. 2.3, the byte organization of Shadow-512's state is shown according to the reference C code of Shadow-512. Each number indicates the index of the byte input in the array.

B_0			
15	14	13	12
11	10	9	8
7	6	5	4
3	2	1	0

B_1			
31	30	29	28
27	26	25	24
23	22	21	20
19	18	17	16

B_2			
47	46	45	44
43	42	41	40
39	38	37	36
35	34	33	32

B_3			
63	62	61	60
59	58	57	56
55	54	53	52
51	50	49	48

Figure 2.3: State of Shadow-512

Each bundle consists of four 32-bit words and has 32 S-Boxes. The bit organization of a bundle is shown in Fig. 2.4. s_k^j denotes a bit. k denotes the row index for $(0 \leq k \leq 3)$, j denotes the column index for $(0 \leq j \leq 31)$. s_3^{31} denotes the most significant bit of the S-Box s^{31}.

s^{31}	s^{30}	s^{29}	...	s^{2}	s^{1}	s^{0}
s_3^{31}	s_3^{30}	s_3^{29}		s_3^{2}	s_3^{1}	s_3^{0}
s_2^{31}	s_2^{30}	s_2^{29}	...	s_2^{2}	s_2^{1}	s_2^{0}
s_1^{31}	s_1^{30}	s_1^{29}		s_1^{2}	s_1^{1}	s_1^{0}
s_0^{31}	s_0^{30}	s_0^{29}		s_0^{2}	s_0^{1}	s_0^{0}

Figure 2.4: Bit organization of a Shadow-512's bundle

2.4. Components of Clyde-128 and Shadow-512

Clyde-128 and Shadow-512 use the same S-Box, L-Box and round constant addition part in their LS-designs. Since Shadow-512 has a multiple LS-design, it needs a diffusion layer which is D-Box to mix bundles with each other.

2.4.1. S-Box Layer

S-box is the non-linear part of both Shadow-512 and Clyde-128 designs. It provides confusion. 4x4 S-Box is used in the Shadow-512 and Clyde-128 design. The S-box is the modified S-Box of Skinny (Beierle et al., 2016). NOR gates in the Skinny S-Box are replaced with AND gates. There are 128 S-Boxes in a Shadow-512's state and 32 S-Boxes in a Clyde-128's state. Each column in the state is the input of an S-Box. S-box and inverse S-Box are shown in Table 2.1.

Table 2.1: S-Box and inverse S-Box of Spook in table representation

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

S-Box can also be described with four AND and four XOR operations. The formula of S-Box and inverse S-Box is given below. $y = S(x)$ denotes the S-Box operation. x_3, x_2, x_1, x_0 represent the four 32-bit rows of a Bundle and y_3, y_2, y_1, y_0represent the four 32-bit rows of output.

S-Box Implementation:

$y_1 = (x_0 \odot x_1) \oplus x_2$

$y_0 = (x_3 \odot x_0) \oplus x_1$

$y_3 = (y_1 \odot x_3) \oplus x_0$

$y_2 = (y_0 \odot y_1) \oplus x_3$

S^{-1}-Box Implementation

$y_3 = (x_0 \odot x_1) \oplus x_2$

$y_0 = (x_3 \odot y_3) \oplus x_3$

$y_1 = (y_3 \odot y_0) \oplus x_0$

$y_2 = (y_0 \odot y_1) \oplus x_1$

2.4.2. L-Box Layer

L-Box is the linear part of both Shadow-512 and Clyde-128 designs. Each bundle consists of four 32-bit words. Two L-Boxes are used in a bundle. It means that the first and second rows are the L-Box inputs and also the third and fourth rows are the inputs of another L-Box. L-Box design includes right rotation and XOR operation, which helps prevent cache attacks (Tromer et al., 2010).

The formula of L-Box is given below. The inputs of the L-Box are denoted as x_i and x_{i+1} and the outputs of the L-Box are denoted as y_i and y_{i+1} where $i = 0$ or $i = 2$ for a bundle. x_3,x_2, x_1 and x_0 are considered as the four 32-bit rows of a Bundle and y_3,y_2, y and y_0 are considered as the four 32-bit rows of output. The circulant matrix is denoted by $circ$ and its input is given in the formula below.

$$(y_i, y_{i+1}) = LBox(x_i, x_{i+1}) = \begin{pmatrix} circ(0xec045008) \cdot x_i^T \oplus circ(0x36000f60) \cdot x_{i+1}^T \\ circ(0x1b0007b0) \cdot x_i^T \oplus circ(0xec045008) \cdot x_{i+1}^T \end{pmatrix}$$

In this thesis work and reference C code of Spook, L-Box and inverse L-Box were implemented as word-level right rotations which are denoted as $rotr$ and 32-bit XORs according to Table 2.2.

Table 2.2: L-Box and inverse L-Box of Shadow-512

L-Box	**Inverse L-Box**
$a = x_i \oplus rotr(x_i, 12)$	$a = x_i \oplus rotr(x_i, 25)$
$b = x_{i+1} \oplus rotr(x_{i+1}, 12)$	$b = x_{i+1} \oplus rotr(x_{i+1}, 25)$
$a = a \oplus rotr(a, 3)$	$c = x_i \oplus rotr(a, 31)$
$b = b \oplus rotr(b, 3)$	$d = x_{i+1} \oplus rotr(b, 31)$
$a = a \oplus rotr(x, 17)$	$c = c \oplus rotr(a, 20)$
$b = b \oplus rotr(y, 17)$	$d = d \oplus rotr(b, 20)$
$c = a \oplus rotr(a, 31)$	$a = c \oplus rotr(c, 31)$
$d = b \oplus rotr(b, 31)$	$b = d \oplus rotr(d, 31)$
$a = a \oplus rotr(d, 26)$	$c = c \oplus rotr(b, 26)$
$b = b \oplus rotr(c, 25)$	$d = d \oplus rotr(a, 25)$
$a = a \oplus rotr(c, 15)$	$a = a \oplus rotr(c, 17)$
$b = b \oplus rotr(d, 15)$	$b = b \oplus rotr(d, 17)$
$y_i = a$	$a = rotr(a, 16)$
$y_{i+1} = b$	$b = rotr(b, 16)$
	$y_i = a$
	$y_{i+1} = b$

2.4.3. Round Constant Addition

Clyde-128 and Shadow-512 uses 4-bit round constants that are generated from a 4-bit LFSR. Round constant is added to 0^{th} column, which is s^0 in Clyde-128 design.

For Shadow-512, round constant is added to different columns of different bundles. In other words, the round constant is added to i^{th} column of bundle B_i such that s^0 to B_0, s^1 to B_1, s^2 to B_2, s^3 to B_3.

Shadow-512 and Clyde-128 are designed as six steps. Since each step consists of two rounds: Round A and Round B, there are twelve round constants that are generated from LFSR. In Table 2.3, round constants of each step are shown. The first bit represents the least significant bit.

Table 2.3: Round constants of Shadow-512

Round Constants		
Round-0: (1,0,0,0)	Round-1: (0,1,0,0)	Round-2: (0,0,1,0)
Round-3: (0,0,0,1)	Round-4: (1,1,0,0)	Round-5: (0,1,1,0)
Round-6: (0,0,1,1)	Round-7: (1,1,0,1)	Round-8: (1,0,1,0)
Round-9: (0,1,0,1)	Round-10: (1,1,1,0)	Round-11: (0,1,1,1)

2.4.4. D-Box Layer

D-Box is a diffusion layer that is used to mix four bundles. It is a diffusion part of Shadow-512 permutation. Clyde-128 does not have a D-Box layer. S-Box, L-Box and round constant addition only affect a bundle itself. Due to the D-Box layer, each bundle diffuses to other bundles. D-Box operation is based on a near-MDS matrix which was used in (Banik et al., 2015). The 4×4 matrix of D-Box is given below. B_j denotes a 128-bit bundle for $0 \leq j \leq 3$.

$$(B_0', B_1', B_2', B_3') = D - Box(B_0, B_1, B_2, B_3) = \begin{pmatrix} 0\,1\,1\,1 \\ 1\,0\,1\,1 \\ 1\,1\,0\,1 \\ 1\,1\,1\,0 \end{pmatrix} \times \begin{pmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{pmatrix}$$

In this thesis and in the reference C code of the Spook, D-Box is implemented according to the formula given below.

- $B_0' = B_1 \oplus B_2 \oplus B_3$
- $B_1' = B_0 \oplus B_2 \oplus B_3$
- $B_2' = B_0 \oplus B_1 \oplus B_3$
- $B_3' = B_0 \oplus B_1 \oplus B_2$

According to the formulas above, it is obvious that three of four bundles are XORed with each other to construct the other bundle. It means, Bundle-0, Bundle-1 and Bundle-2 are XORed to obtain Bundle-3, Bundle-0, Bundle-1 and Bundle-3 are

XORed to obtain Bundle-2. Bundle-0, Bundle-2 and Bundle-3 are XORed to obtain Bundle-1. Bundle-1, Bundle-2 and Bundle-3 are XORed to obtain Bundle-0. Since near-MDS 4×4 matrix is an involutory matrix which is $D^2 = I$, inverse D-Box operation is identical to D-Box operation. Therefore, the formulas given above are valid for inverse D-Box operation.

2.5. Security Claims of Spook

In (Bellizia et al., 2019), the authors of Spook stated the security of the algorithm based on three main components, which are S1P mode of operation, The Clyde-128 TBC and Shadow-512 permutation. They stated that the security of the S1P mode of operation relies on the assumption that the secret key of TBC cannot be leaked. In addition, the secret key is only used twice in both the encryption and decryption process. They referred to (C. Guo et al., 2019) for proof of their assumptions. They performed the linear and differential attacks on Clyde-128 by using the wide-trail strategy (Daemen & Rijmen, 2001). They implied that the best linear/differential characteristics were found with the probability 2^{-128} after four steps. Since Clyde-128 is designed as six steps, they believed that the Clyde-128 is resistant against linear/differential attacks. Also, they referred to (Boura et al., 2011) for the upper bound of the algebraic degree and they stated that five rounds (two and a half steps) are enough to reach a maximum algebraic degree. Therefore, they believed that the six steps of Clyde-128 provide security against algebraic attacks (Courtois & Pieprzyk, 2002), cube attacks (Dinur & Shamir, 2009) and division property which is a property of integral attacks (Todo, 2015). For division property, they used the MILP method (Xiang et al., 2016), a tool to search integral distinguishers and found a 4-Step integral distinguisher of Clyde-128. From the point of invariant attacks, which was previously considered to be related to the S-Box (J. Guo et al., 2016), they chose the round constants of Clyde-128 and Shadow-512 by taking into consideration of (Beierle et al., 2017). Since the round constants are invariant parts of the linear layer and have a major role against invariant attacks, they wanted to increase the dimension of the invariant subspaces of the linear layer. To provide security against the chosen-tweak attack for Clyde-128 TBC, they determined the minimum number of rounds required and doubled the number of rounds by using the approach in (J. Guo et al., 2011). For subspace trail analysis introduced in (Grassi et al., 2017), the authors used algorithmic approach which was introduced in (Leander et al., 2018) to compute the best subspace trails for Clyde-128. They stated that they found one and a half step subspace trails for Clyde-128. For Shadow-512 permutation, since they wanted to obtain better performance results, they did not aim to meet strong security properties from Shadow-512. They stated that they obtained the upper bound of 2^{-128} for linear characteristics after two steps. For the tag generation, 255 bits are used and Shadow-512 permutation should provide collision resistance for 255 bits. The authors said that the best truncated differential characteristics of Shadow-512 cannot be found with the probability better than 2^{-385} after six steps.

The authors of Spook also stated that Shadow-512 could be used as four steps instead of six steps. It can be said that this statement is the motivation of (Derbez et al., 2020) and this thesis. In (Derbez et al., 2020), they found 5-Step, 6-Step and 7-Step

truncated differential distinguishers. Moreover, they performed a 4-Step forgery attack. Their truncated differentials will be explained in detail in Chapter 3. In this thesis, we found 6, 7-Step impossible differential distinguisher of Shadow-512. In addition, we obtained 8-Step improbable differential distinguisher of Shadow-512, which is the largest distinguisher that is provided for this permutation.

CHAPTER 3

CRYPTANALYSIS RESULT ON SPOOK BRINGING FULL-ROUND SHADOW-512 TO THE LIGHT

In Chapter 2, the Spook (Bellizia et al., 2019) algorithm and its components were explained. As mentioned in Section 2.3, Spook uses an SPN permutation whose name is Shadow-512. In this chapter, the cryptanalysis result of Shadow-512 that was introduced in (Derbez et al., 2020) will be explained. They found practical distinguishers of Shadow-512 permutation. They presented a truncated differential distinguisher with probability one that covers five steps of Shadow-512. This 5-Step distinguisher was used to find our impossible and improbable distinguishers that will be explained in Chapter 4. They also found a second truncated differential distinguisher that covers six steps of Shadow-512. In addition, they showed that there is a distinguisher that covers seven steps as if it is a round-extended version of Shadow by adding one more round at the bottom of the 6-Step truncated differential. They said that all these distinguishers are practical and can be verified experimentally. They also performed a forgery attack against four steps of Shadow-512 in a nonce-misuse scenario. They stated that they could obtain collisions on four steps Shadow-512 by using the nonce three times allowed by (Berti et al., 2017). However, they could not use their 5-Step, 6-Step, and 7-Step truncated differential distinguishers on forgery attacks because the S1P mode of operation does not allow them to control capacity bits in the input. In this chapter, firstly, preliminary knowledge about the construction of truncated trails will be given. The concepts of Super S-Box and 3-Identical state will be explained. Secondly, 5-Step truncated differential distinguisher will be shown. Thirdly, 6-Step and 7-Step truncated differential distinguishers will be described.

Table 3.1: Notation of Chapter 3

B_j	$128-$ bit Bundle, for $0 \leq j \leq 3$
p_j	Probability
σ_j	$128-$ bit Super S $-$ Box of Bundle j
$S(x)$	S $-$ Box operation on x
$L(x)$	L $-$ Box operation on x
$D(x)$	D $-$ Box operation on x
$x \oplus y$	Bitwise XOR of $128-$ bit Bundles x and y
Δ_i	The difference of two inputs of Bundle i
α, β, ϕ	The difference value of $128-$ bit Bundle.
u^j	j^{th} column of State u.
π	Shadow $-$ 512 permutation.

3.1. Preliminaries of Distinguishers

In this section, the base elements that help the authors find truncated differentials will be introduced. Firstly, the Super S-Box structure will be described. Then, the 3-Identical State and its usage will be explained for Shadow-512 permutation.

3.1.1. Super S-Box

Shadow-512 permutation is considered as SPN. As mentioned in Chapter 2, each step consists of two rounds which are Round A and Round B. Round A comprises S-Box, L-Box and round constant addition parts. Round B contains S-Box, D-Box and round constant addition parts. In (Derbez et al., 2020), they introduce a new layer which is called Super S-Box, by separating the D-Box layer from the other layers. Since only the D-Box layer mixes the bundles and the other layers affect only a 128-bit bundle, each step can be represented by a Super S-Box and D-Box layer. To conclude, 512-bit Shadow state uses four 128-bit Super S-Boxes and one 512-bit D-Box within a step.

3.1.2. 3-Identical State

3-Identical state means that inputs of three of four bundles are set to the same value before the Super S-Box operation of a step. In other words, the same 128-bit value is used for three bundles in a state. Normally, it is expected that the 3-Identical state cannot be preserved after the first round constant addition part. As mentioned in Section 2.4.3, the round constant is added to different columns for different bundles. Some conditions must be satisfied to keep the 3-Identical State after a Super S-Box and D-Box operation. Let Υ be 512-bit 3-Identical Shadow state such that $\Upsilon = (u, u, u, t)$. Since Bundle-0, Bundle-1 and Bundle-2 are in the 3-Identical state, they will keep the 3-Identical state after the S-Box and L-Box operation. Therefore, the state will be $L(S(\Upsilon)) = (v, v, v, z)$ after L-Box.

The State after L-Box:

B_0: v^{31}, . . . v^3, v^2, v^1, v^0

B_1: v^{31}, . . . v^3, v^2, v^1, v^0

B_2: v^{31}, . . . v^3, v^2, v^1, v^0

B_3: z^{31}, . . . z^3, z^2, z^1, z^0

The State after round constant addition: Round constant c is added according to bundle index.

B_0: v^{31}, . . . v^3, v^2, v^1, $v^0 \oplus c$

B_1: v^{31}, . . . v^3, v^2, $v^1 \oplus c$, v^0

B_2: v^{31}, . . . v^3, $v^2 \oplus c$, v^1, v^0

B_3: z^{31}, . . . $z^3 \oplus c$, z^2, z^1, z^0

The State after second S-Box:

B_0: $S(v^{31})$, . . . $S(v^3)$, $S(v^2)$, $S(v^1)$, $S(v^0 \oplus c)$

B_1: $S(v^{31})$, . . . $S(v^3)$, $S(v^2)$, $S(v^1 \oplus c)$, $S(v^0)$

B_2: $S(v^{31})$, . . . $S(v^3)$, $S(v^2 \oplus c)$, $S(v^1)$, $S(v^0)$

B_3: $S(z^{31})$, . . . $S(z^3 \oplus c)$, $S(z^2)$, $S(z^1)$, $S(z^0)$

The State after D-Box: Three of four bundles are XORed with each other.

B_0: $S(z^{31}), \ldots\ S(z^3 \oplus c), S(v^2) \oplus S(v^2 \oplus c) \oplus S(z^2), \quad S(v^1) \oplus S(v^1 \oplus c) \oplus S(z^1), \quad S(z^0)$

B_1: $S(z^{31}), \ldots\ S(z^3 \oplus c), \quad S(v^2) \oplus S(v^2 \oplus c) \oplus S(z^2), \quad S(z^1), \quad S(v^0) \oplus S(v^0 \oplus c) \oplus S(z^0)$

B_2: $S(z^{31}), \ldots\ S(z^3 \oplus c), \quad S(z^2), \quad S(v^1) \oplus S(v^1 \oplus c) \oplus S(z^1), \quad S(v^0) \oplus S(v^0 \oplus c) \oplus S(z^0)$

B_3: $S(z^{31}), \ldots\ S(v^3), \quad S(v^2 \oplus c), \quad S(v^1 \oplus c), \quad S(v^0 \oplus c)$

The State after round constant addition: Round constant c' is added according to bundle index.

B_0: $S(z^{31}), \ldots\ S(z^3 \oplus c), \quad S(v^2) \oplus S(v^2 \oplus c) \oplus S(z^2), \quad S(v^1) \oplus S(v^1 \oplus c) \oplus S(z^1), \quad S(z^0) \oplus c'$

B_1: $S(z^{31}), \ldots\ S(z^3 \oplus c), \quad S(v^2) \oplus S(v^2 \oplus c) \oplus S(z^2), \quad S(z^1) \oplus c', \quad S(v^0) \oplus S(v^0 \oplus c) \oplus S(z^0)$

B_2: $S(z^{31}), \ldots\ S(z^3 \oplus c), \quad S(z^2) \oplus c', \quad S(v^1) \oplus S(v^1 \oplus c) \oplus S(z^1), \quad S(v^0) \oplus S(v^0 \oplus c) \oplus S(z^0)$

B_3: $S(z^{31}), \ldots\ S(v^3) \oplus c', \quad S(v^2 \oplus c), \quad S(v^1 \oplus c), \quad S(v^0 \oplus c)$

As can be seen that to keep the 3-Identical state after one step following equations should be satisfied:

- $S(v^2) \oplus c' = S(v^2 \oplus c)$ (3.1)
- $S(v^1) \oplus c' = S(v^1 \oplus c)$ (3.2)
- $S(v^0) \oplus c' = S(v^0 \oplus c)$ (3.3)

The exact value of round constants c and c' are known for each step index. Therefore, it is possible to satisfy the equations (3.1), (3.2) and (3.3) for some steps with some probability. The authors of (Derbez et al., 2020) showed that if the input is 3-Identical, the probability of obtaining 3-Identical output after one step, according to Table 3.2. The probabilities do not depend on the bundle indices.

Table 3.2: The probability of 3-Identical output when inputs are 3-Identical

Step No:	0	1	2	3	4	5
Probability:	0	0	0	2^{-9}	2^{-6}	0

The analysis shows that satisfying these three equations depends on the round constants that are used in the step and the round constants depend on the step index. This type of analysis is known as exploiting the sparse round constants. There are some previous examples of exploiting sparse round constants. They are used in rotational cryptanalysis (Khovratovich et al., 2015), differential attacks (Peyrin, 2010), self-similarity (Bouillaguet et al., 2010) and invariant subspace attacks (Leander et al., 2011, 2015).

3.2. 5-Step Truncated Differential Distinguisher of Shadow-512

In (Derbez et al., 2020), they found 5-Step truncated differential distinguisher of Shadow-512 permutation with probability one. The 5-Step truncated differential distinguisher can be considered as a rebound attack (Mendel et al., 2009). They found a distinguisher by exploiting D-Box. The 5-Step truncated differential that is shown in Fig. 3.1 starts from Step-2 with the 3-Identical state and two steps are applied both forward and backward direction with probability one. Algorithm-1 summarizes the 5-Step distinguisher. In this thesis, we found 6-Step and 7-Step impossible differential distinguishers and 8-Step improbable differential distinguisher of Shadow-512 permutation by using this 5-Step truncated differential. Therefore, it can be considered as the starting point of this thesis work.

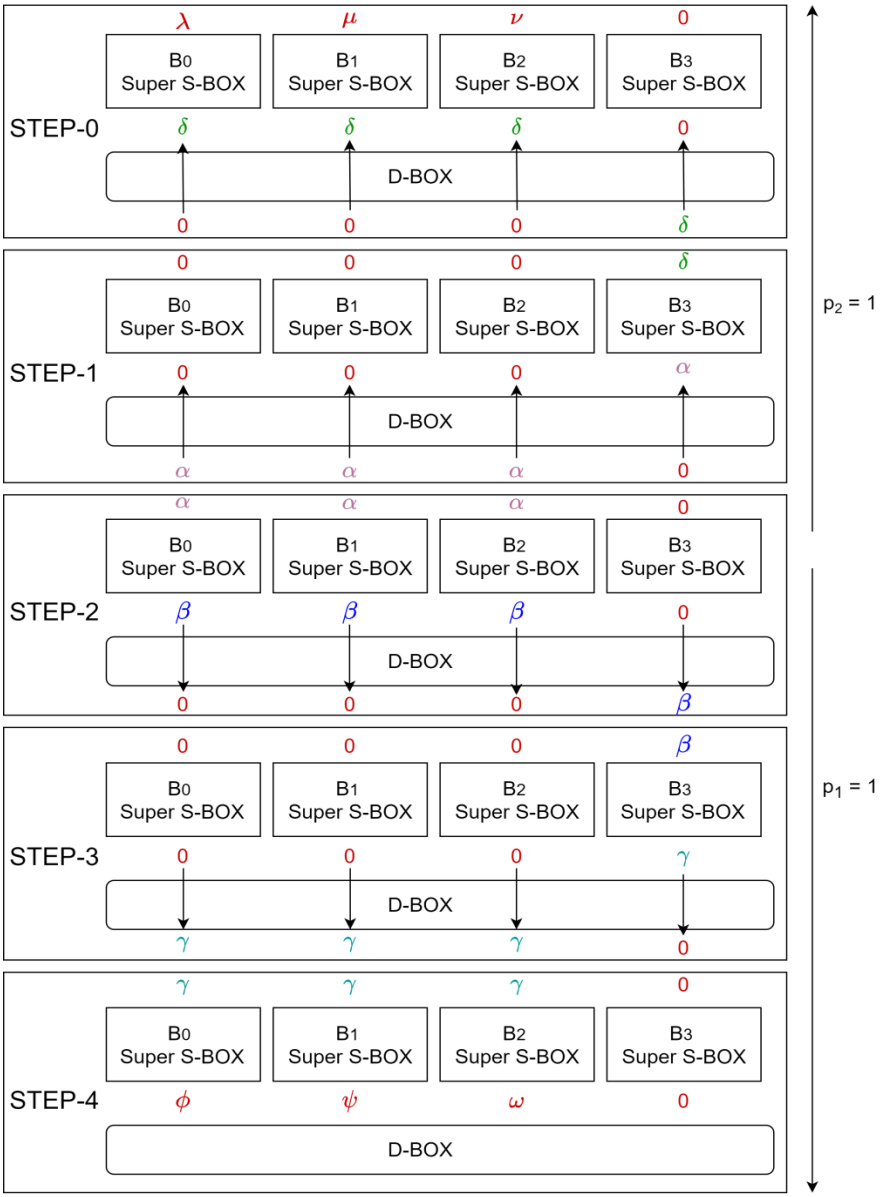

Figure 3.1: 5-Step Truncated Differential of Shadow-512

Algorithm − 1 5 − Step distinguisher with probability one

i. Choose a random pair such that $v \in \mathbb{F}_2^{128}$ and $(v \oplus \beta) \in \mathbb{F}_2^{128}$ in Step-2. β must be set to zero on the 0^{th}, 1^{st},2^{nd},3^{rd} columns of a bundle.

ii. Choose a random state $z \in \mathbb{F}_2^{128}$.

iii. Compute $u = \sigma_j^{-1}(v)$ and $u + \alpha = \sigma_j^{-1}(v + \beta)$ for $0 \leq j \leq 2$. Set the states at Step-2 such that

$$X_2 = (u, u, u, z) \; and \; X_2' = (u + \alpha, u + \alpha, u + \alpha, z) \rightarrow \Delta = (\alpha, \alpha, \alpha, 0)$$

iv. Iterate Step-4 and Step-5 on X_2 and X_2' to obtain $(\phi, \psi, \omega, 0)$ in Step-5

According to Algorithm-1, the 5-Step truncated differential starts from Step-2 with the 3-Identical state. In other words, the input pairs of Bundle-0, Bundle-1 and Bundle-2 are chosen the same, so their difference values become $(\alpha, \alpha, \alpha, 0)$. The purpose is to obtain three identical differences for Bundle-0, Bundle-1 and Bundle-2 just before the D-Box layer of Step-2. Super S-Boxes are slightly different on columns that round constants are added. For a bundle index *j*, the round constant is added to j^{th} column of a bundle. Therefore, 3-Identical State may not be preserved after the round constant addition part as mentioned in Section 3.2.2. However, the differences of Bundle-0, Bundle-1 and Bundle-2 must still be the same because round constant addition is just an XOR operation that does not change the difference. The impact of the round constant addition is observed in the second S-Box layer that is done right after the round constant addition. Since round constants are added to different columns of different bundles, the input of the S-Boxes will be different. The same input difference can lead to different output differences in the S-Box layer. Therefore, even if they start with three identical input differences in Step-2, they may not obtain three identical output differences at the end of Super S-Box. To achieve this, they choose the input difference α so that α does not diffuse to columns that round constants are added and the output difference will be β. It is obvious that β must be set to zero on columns that round constants are added. The proof of this approach is explained below.

Let u and $(u \oplus \alpha)$ be two inputs for Bundle-0, Bundle-1 and Bundle-2 at the beginning of Step-2. After the L-Box layer, the output difference and output pairs are still the same for Bundle-0, Bundle-1 and Bundle-2 because the same operation is applied to the same bundles.

$LBox\big(SBox(u)\big) = v$ and $LBox\big(SBox(u \oplus \alpha)\big) = v \oplus \varepsilon$

$v = v^{31} \ldots v^4 \; v^3 \; v^2 \; v^1 \; v^0$

$v \oplus \beta = v^{31} \oplus \varepsilon \ldots v^4 \oplus \varepsilon \; v^3 \; v^2 \; v^1 \; v^0$

After the L-Box layer, round constants are added to bundles according to their bundle index. Then S-Box operation is applied.

$\sigma_0(u) = S(v^{31}) \ldots S(v^4) \; S(v^3) \; S(v^2) \; S(v^1) \; \boldsymbol{S(v^0 \oplus c)}$

$\sigma_0(u \oplus \alpha) = S(v^{31} \oplus \varepsilon) \ldots S(v^4 \oplus \varepsilon) \; S(v^3) \; S(v^2) \; S(v^1) \; \boldsymbol{S(v^0 \oplus c)}$

$\sigma_1(u) = S(v^{31}) \ldots S(v^4) \; S(v^3) \; S(v^2) \; \boldsymbol{S(v^1 \oplus c)} \; S(v^0)$

$\sigma_1(u \oplus \alpha) = S(v^{31} \oplus \varepsilon) \ldots S(v^4 \oplus \varepsilon) \; S(v^3) \; S(v^2) \; \boldsymbol{S(v^1 \oplus c)} \; S(v^0)$

$\sigma_2(u) = S(v^{31}) \ldots S(v^4) \; S(v^3) \; \boldsymbol{S(v^2 \oplus c)} \; S(v^1) \; S(v^0)$

$\sigma_2(u \oplus \alpha) = S(v^{31} \oplus \varepsilon) \ldots S(v^4 \oplus \varepsilon) \; S(v^3) \; \boldsymbol{S(v^2 \oplus c)} \; S(v^1) \; S(v^0)$

The differences of Bundle-0 and Bundle-1 and Bundle-2 are identical after the Super S-Box layer:

$$\Delta_0 = \sigma_0(u) \oplus \sigma_0(u \oplus \alpha) = \quad k^{31} \quad \ldots \quad k^4 \quad 0 \quad 0 \quad 0 \quad 0 = \beta$$

$$\Delta_1 = \sigma_1(u) \oplus \sigma_1(u \oplus \alpha) = \quad k^{31} \quad \ldots \quad k^4 \quad 0 \quad 0 \quad 0 \quad 0 = \beta$$

$$\Delta_2 = \sigma_1(u) \oplus \sigma_1(u \oplus \alpha) = \quad k^{31} \quad \ldots \quad k^4 \quad 0 \quad 0 \quad 0 \quad 0 = \beta$$

In a nutshell, if the output difference of β is set to zero on columns that round constants are added, 3-Identical State leads to three identical output differences after the Super S-Box layer. Therefore, the input difference of Step-2 which is $(\alpha, \alpha, \alpha, 0)$ will become $(\beta, \beta, \beta, 0)$ after Super S-Box layer.

As mentioned in Section 2.4.4, three of four bundles are XORed with each other to build the other bundle in the D-Box layer. In other words, the D-Box layer mixes the bundles to provide diffusion. After the D-Box layer of Step-2, the difference $(\beta, \beta, \beta, 0)$ will become $(0,0,0, \beta)$.In Step-3, the difference $(0,0,0, \beta)$ will become $(0,0,0, \gamma)$. In the D-Box layer of Step-3, Bundle-3 difference, which is γ will diffuse to the other bundles and the difference will become $(\gamma, \gamma, \gamma, 0)$. The differences of Bundle-0, Bundle-1 and Bundle-2 are equal at the beginning of Step-4; however, they are not in 3-Identical State. Therefore, after the Super S-Box layer of Step-4, their differences will be different from each other. It will be $(\phi, \psi, \omega, 0)$. The difference values of Bundle-0, Bundle-1 and Bundle-2 will be different from each other and the difference of Bundle-3 will be definitely zero after the Super S-Box of Step-4 with probability one. Now, we are going back to Step-2 to apply two steps inverse operation in the backward direction. The difference $(\alpha, \alpha, \alpha, 0)$ will become $(0,0,0, \alpha)$ after the inverse D-Box layer of Step-1. Then inverse S-Box operation is applied. The difference at the beginning of Step-1 will be $(0,0,0, \delta)$. The difference of Bundle-3 will diffuse to other bundles in the inverse D-Box of Step-0. The difference value will become $(\delta, \delta, \delta, 0)$. In Step-0, the last operation, which is inverse S-Box is applied to bundles and the difference $(\lambda, \mu, \nu, 0)$ is obtained. The difference value of Bundle-0, Bundle-1 and Bundle-2 will be different from each other, but the difference of Bundle-3 will be definitely zero at the beginning of Step-0. In a nutshell, if the difference of Step-2 is $(\beta, \beta, \beta, 0)$ and β is set to zero on the first four columns, the difference of Bundle-3 will be zero in both Step-0 and Step-5 with probability one.

To sum up, in (Derbez et al., 2020), they found a practical distinguisher that covers five steps of Shadow-512 permutation with probability one. The truncated trail starts from Step-2 with the 3-Identical state; however, the 3-Identical state is not preserved at the beginning of Step-3. In Step-4, the difference will be $(\phi, \psi, \omega, 0)$. The difference values of Bundle-0, Bundle-1, and Bundle-2 are different from each in Step-4. Therefore, the authors stated that the truncated trail could not cover more than five steps with probability one. They also noted that the 5-Step truncated differential distinguisher was verified experimentally. It can be used to distinguish the five steps of Shadow-512 from a random permutation by using a pair of inputs.

3.3. 6- Step and 7-Step Truncated Differential Distinguisher of Shadow-512

In (Derbez et al., 2020), they also found 6-Step and 7-Step truncated differential distinguisher by using their 5-Step truncated differential distinguisher that was explained in the previous section. 6-Step truncated differential distinguisher covers full permutation and it can be extended to distinguish 7-Step of Shadow-512, which is considered as round-extended version. Figure 3.2 shows the 7-Step truncated differential distinguisher.

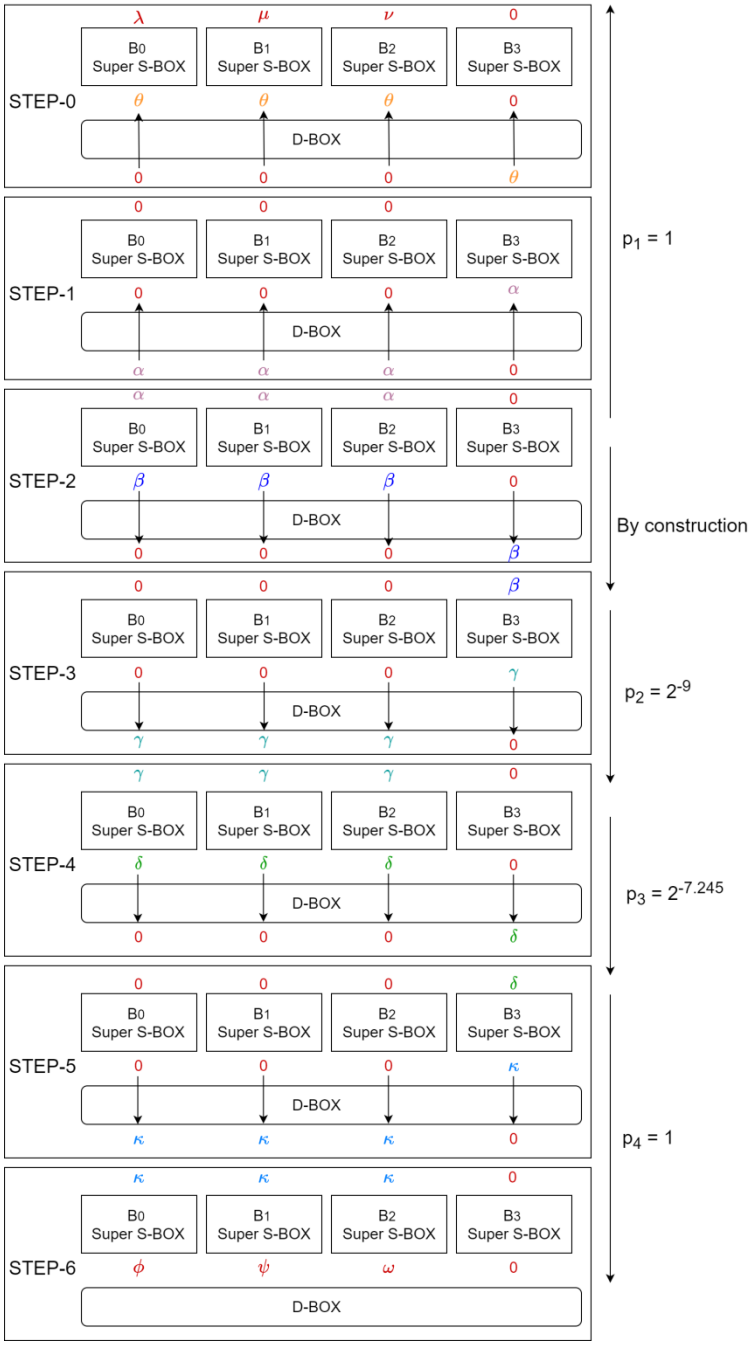

Figure 3.2: 7-Step Truncated Differential of Shadow-512

Let U and U' be 512-bit Shadow-512 input messages. They found a distinguisher for 6-Step Shadow-512, such as

$U \oplus U' = (\lambda, \mu, \nu, 0)$ and $\pi(U) \oplus \pi(U') = DBox(0,0,0,\kappa)$ with probability $2^{-16.245}$.

Also, they found a distinguisher for 7-Step Shadow-512, such as

$U \oplus U' = (\lambda, \mu, \nu, 0)$ and $\pi(U) \oplus \pi(U') = DBox(\phi, \psi, \omega, 0)$ with probability $2^{-16.245}$.

Algorithm-2 summarizes the 7-Step truncated differential distinguisher.

Algorithm − 2 7 − Step distinguisher

i. Choose a difference $\alpha \in \mathbb{F}_2^{128}$. Set α to zero except for 22nd and 23rd columns.

ii. Choose a pair of state $(u_3, u_3, u_3, w_3)$and $(u_3, u_3, u_3, w_3 \oplus \beta)$ at the beginning of Step-3.

iii. Invert Step-2 on (u_3, u_3, u_3, w_3) and $(u_3, u_3, u_3, w_3 \oplus \beta)$ to obtain (k_2, l_2, m_2, n_2) and $(k_2 \oplus \alpha, l_2 \oplus \alpha, m_2 \oplus \alpha, n_2)$

iv. Invert Step-1 and Step-0 on (k_2, l_2, m_2, n_2) and $(k_2 + \alpha, l_2 + \alpha, m_2 + \alpha, n_2)$ to obtain zero difference in Bundle-3 such that $\Delta_0 = (k_0, l_0, m_0, n_0) \oplus (k_0', l_0', m_0', n_0') = (x, y, z, 0)$

v. Return this pair to satisfy the truncated trail with the probability of $2^{-16.24}$.

Before starting the truncated trail, firstly, a pair of inputs should be built in Step-2. At the beginning of Step-2, pairs of bundles are not in a 3-Identical state. However, pairs of Step-2 are built so that they should be in the 3-Identical state at the beginning of Step-3. The input difference of Step-2 $(\alpha, \alpha, \alpha, 0)$ should be $(\beta, \beta, \beta, 0)$ after Super S-Box and the difference $(\beta, \beta, \beta, 0)$ should be $(0,0,0,\beta)$ at the end of Step-2. Also, the state of Step-3 should be 3-Identical. To do that, the difference α is set on only 22nd and 23rd columns. This is because the columns whose indices 22 and 23 do not affect the output columns whose indices 0,1,2,3. In other words, the α difference does not diffuse to the columns that round constant is added and the differences of these columns will be zero after Super S-Box operation. The authors stated that 2^{16} pairs could be generated in Step-3 to satisfy the difference condition of Step-2.

Step-3 starts with the difference $(0,0,0,\beta)$. Also, it is known that the first three bundles are 3-Identical. In Section 3.2.2, three equations were shown to keep the 3-Identical state if the input is 3-Identical. According to Table 3.2, the probability of obtaining 3-Identical output in Step-3 is 2^{-9} if inputs of Step-3 are 3-Identical.

Step-4 starts with the difference $(\gamma, \gamma, \gamma, 0)$. Also, Bundle-0, Bundle1 and Bundle-2 are in the 3-Identical state. The idea is to obtain the output difference $(0,0,0,\delta)$ at the beginning of Step-5 and δ must be a nonzero value.

Let us denote two states of Step-4 after the L-Box operation, such as (v, v, v, z) and (v', v', v', z). After L-Box operation, firstly round constant is added, then S-Box operation is applied. After the S-Box operation, since two states are in the 3-Identical state, Bundle-0, Bundle-1 and Bundle-2 will be the same except the columns that round constants are added. At the end of Step-4, D-Box and second round constant addition are applied. Obviously, the first three bundles only differ by the 0^{th}, 1^{st}, 2^{nd} and 3^{rd} columns. The expression of 0^{th}, 1^{st}, 2^{nd} and 3^{rd} columns of (v, v, v, z) and (v', v', v', z) at the end of Step-4 is shown below. c denotes the first round constant and c' denotes the second round constant.

- For (v, v, v, z):

B_0: $S(z^4 \oplus c)$, $S(v^2) \oplus S(v^2 \oplus c) \oplus S(z^2)$, $S(v^1) \oplus S(v^1 \oplus c) \oplus S(z^1)$, $S(z^0) \oplus c'$

B_1: $S(z^3 \oplus c)$, $S(v^2) \oplus S(v^2 \oplus c) \oplus S(z^2)$, $S(z^1) \oplus c'$, $S(v^0) \oplus S(v^0 \oplus c) \oplus S(z^0)$

B_2: $S(z^3 \oplus c)$, $S(z^2) \oplus c'$, $S(v^1) \oplus S(v^1 \oplus c) \oplus S(z^1)$, $S(v^0) \oplus S(v^0 \oplus c) \oplus S(z^0)$

B_3: $S(v^3) \oplus c'$, $S(v^2 \oplus c)$, $S(v^1 \oplus c)$, $S(v^0 \oplus c)$

- For (v', v', v', z):

B'_0: $S(z^4 \oplus c)$, $S(v'^2) \oplus S(v'^2 \oplus c) \oplus S(z^2)$, $S(v'^1) \oplus S(v'^1 \oplus c) \oplus S(z^1)$, $S(z^0) \oplus c'$

B'_1: $S(z^3 \oplus c)$, $S(v'^2) \oplus S(v'^2 \oplus c) \oplus S(z^2)$, $S(z^1) \oplus c'$, $S(v'^0) \oplus S(v'^0 \oplus c) \oplus S(z^0)$

B'_2: $S(z^3 \oplus c)$, $S(z^2) \oplus c'$, $S(v'^1) \oplus S(v'^1 \oplus c) \oplus S(z^1)$, $S(v'^0) \oplus S(v'^0 \oplus c) \oplus S(z^0)$

B'_3: $S(v'^3) \oplus c'$, $S(v'^2 \oplus c)$, $S(v'^1 \oplus c)$, $S(v'^0 \oplus c)$

To obtain the difference $(0,0,0,\delta)$ after Step-4, $B_j = B'_j$ for $0 \leq j \leq 3$ must be satisfied.

$$S(v'^2) \oplus S(v'^2 \oplus c) = S(v^2) \oplus S(v^2 \oplus c)$$

$$S(v'^1) \oplus S(v'^1 \oplus c) = S(v^1) \oplus S(v^1 \oplus c)$$

$$S(v'^0) \oplus S(v'^0 \oplus c) = S(v^0) \oplus S(v^0 \oplus c)$$

Since these relations are obtained in Step-4, the round constant is $c = 0x5$. The authors stated that the probability of satisfying all three relations is $2^{-7.245}$.

Step-5 starts with the difference $(0,0,0,\delta)$. The input difference of Step-5 will become $(0,0,0,\kappa)$ after the Super S-Box. The difference value of Bundle-3 κ will diffuse to other bundles and the output difference of Step-5 will be $(\kappa, \kappa, \kappa, 0)$. This is the 6-Step truncated differential distinguisher with the probability $2^{-9} \times 2^{-7.245} = 2^{-16.245}$. Naturally, the six steps can be extended to seven steps by adding an additional step with probability one. After the Super S-Box of Step-6, the output difference will become $(\phi, \psi, \omega, 0)$ with probability one. Therefore, the probability of having zero Bundle-3 difference at the end of Step-6 is also $2^{-16.245}$, if pairs of Step-2 are constructed the way that they are explained. In (Derbez et al.,

2020), they also verified these distinguishers experimentally. They stated that they ran Algorithm-2 for 2^{22} pairs and acquired 124 pairs. The probability of having zero output difference in Bundle-3 is approximately 2^{-15}. Normally, to obtain such an output difference for a random permutation, one needs 2^{64} queries according to (Iwamoto et al., 2013). However, they can use 2^{15} pairs to obtain these differentials for 7-Step Shadow-512.

3.4. Summary of Truncated Differential Distinguishers

In (Derbez et al., 2020), it was shown that there is a 5-Step truncated differential with probability one. Also, they found a distinguisher on full permutation, which is six steps of Shadow-512 with probability $2^{-16.245}$. Naturally, the 6-Step distinguisher can be extended to seven steps with the same probability as if Shadow-512 is designed as seven steps. They stated that these distinguishers are practical and they verified experimentally by using their C++ implementations. In addition, (Derbez et al., 2020) performed a forgery attack on 4-Step Shadow-512. They managed to generate the same tag for two different messages by using the same nonce three times. They used 2^{30} messages and found 41 collisions. However, they could not use their truncated differential distinguishers that are explained in this chapter on the forgery attack because the S1P mode of operation prevents them from specifying the capacity bits which correspond to Bundle-2 and Bundle-3 bits. Therefore their forgery attack starts from Step-2 and covers four steps.

It can be said that it was one of the most comprehensive works that were performed on Spook. Especially, their 5-Step truncated differential distinguisher is one of the main subjects of this thesis work. Since their 5-Step differential works with probability one, we found impossible differential distinguishers by adding one or more rounds top and the bottom of this truncated differential. After the results of (Derbez et al., 2020), the authors of Spook proposed a second version of Spook, Spook v2 (Bellizia et al., 2020). They suggested updating the D-Box layer and round constants. However, Spook v2 was not considered as a round 2 candidate of NIST's competition because Spook v1 is the candidate of the round 2 of the competition. Besides, Spook v1 was not obsoleted by their designers.

CHAPTER 4

IMPOSSIBLE AND IMPROBABLE DIFFERENTIAL DISTINGUISHERS OF SHADOW-512

As mentioned in Chapter 2, Spook is an authenticated encryption algorithm that uses the S1P mode of operation. The S1P mode of operation uses Shadow-512 permutation. In Chapter 2, the design specifications and components of Shadow-512 were explained in detail. In this chapter, firstly, 4×4 bit S-Boxes of Shadow-512 will be investigated. Difference Distribution Table (DDT) and Undisturbed Bits of S-Boxes will be introduced. Secondly, four different distinguishers that distinguish Shadow-512 output from a random permutation will be shown. Shadow-512 was designed as 6-Step. The designers of Spook also recommend that Shadow-512 can be used as 4-Step. In Spook (Bellizia et al., 2019), they stated that the 4-Step design of Shadow-512 is an interesting target for cryptanalysis. Starting from this, we found two different 6-Step impossible differential distinguishers and they will be explained in Section 4.3. Moreover, if Shadow-512 is considered as 7-Step or 8-Step, it is possible to find a distinguisher that covers more steps than 6-Step. In Section 4.4, the 7-Step impossible differential distinguisher and in Section 4.5, 8-Step improbable differential distinguisher will be introduced.

Table 4.1: Notation of Chapter 4

B_j	$128-$ bit Bundle, for $0 \leq \mathrm{j} \leq 3$
p_j	Probability
σ_j	$128-$ bit Super $\mathrm{S}-$ Box of Bundle j
?	A bit difference that can be 1 or 0.
Δ_o, Δ_i	Output difference and Input difference
$x \oplus y$	Bitwise XOR of $128-$ bit Bundles x and y
Δ_i^x	The difference of Bundle i after the operation x
$\mathrm{A}, \mathrm{B}, \alpha, \phi$	The difference value of $128-$ bit Bundle.
$P_{i,j}$	One of the Bundle pair, i denotes the Bundle index, j denotes the pair index

4.1. DDT of S-Box and inverse S-Box.

4×4 bit S-Box is a non-linear part of the Shadow-512 permutation. Since it is a non-linear operation, the output difference can be found with some probability by using the Difference Distribution Table. The Difference Distribution Table shows that which input difference of S-Box leads to which output difference of S-Box for how many times. In other words, the number of every possible input difference is found and the number of their corresponding output differences are counted. Firstly every possible input pairs are XORed with each other such that $x \oplus y = i$. Then, their output pairs are XORed such that $S\text{-}Box(x) \oplus S\text{-}Box(y) = j$. The table is constructed by counting the j values as ij-th entry. DDT of Shadow-512's S-Box is shown in Table 4.2, and DDT of Shadow-512's inverse S-Box is shown in Table 4.3.

Table 4.2: DDT of S-Box of Shadow-512

		Output Difference of S-Box															
		0_x	1_x	2_x	3_x	4_x	5_x	6_x	7_x	8_x	9_x	A_x	B_x	C_x	D_x	E_x	F_x
Input Difference of S-Box	0_x	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1_x	0	0	0	2	0	0	0	2	4	2	0	0	0	2	4	0
	2_x	0	4	0	2	0	4	0	2	0	0	0	2	0	0	0	2
	3_x	0	0	4	0	0	0	0	0	4	2	0	2	0	2	0	2
	4_x	0	0	4	0	0	0	4	0	0	0	4	0	0	0	4	0
	5_x	0	0	0	2	0	0	0	2	4	2	4	0	0	2	0	0
	6_x	0	4	0	2	0	4	0	2	0	0	0	2	0	0	0	2
	7_x	0	0	0	0	0	0	4	0	4	2	0	2	0	2	0	2
	8_x	0	0	0	0	4	4	0	0	0	4	0	0	4	0	0	0
	9_x	0	2	2	0	2	0	0	2	0	0	2	2	2	2	0	0
	A_x	0	0	2	0	0	4	2	0	0	4	2	0	0	0	2	0
	B_x	0	2	0	0	2	0	2	2	0	0	0	2	2	2	2	0
	C_x	0	0	2	2	0	0	2	2	0	0	2	2	0	0	2	2
	D_x	0	2	0	2	2	0	2	0	0	0	0	0	2	2	2	2
	E_x	0	0	0	2	4	0	0	2	0	0	0	2	4	0	0	2
	F_x	0	2	2	2	2	0	0	0	0	0	2	0	2	2	0	2

Table 4.3: DDT of inverse S-Box of Shadow-512

Input Difference of inverse S-Box	Output Difference of inverse S-Box															
	0_x	1_x	2_x	3_x	4_x	5_x	6_x	7_x	8_x	9_x	A_x	B_x	C_x	D_x	E_x	F_x
0_x	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1_x	0	0	4	0	0	0	4	0	0	2	0	2	0	2	0	2
2_x	0	0	0	4	4	0	0	0	0	2	2	0	2	0	0	2
3_x	0	2	2	0	0	2	2	0	0	0	0	0	2	2	2	2
4_x	0	0	0	0	0	0	0	0	4	2	0	2	0	2	4	2
5_x	0	0	4	0	0	0	4	0	4	0	4	0	0	0	0	0
6_x	0	0	0	0	4	0	0	4	0	0	2	2	2	2	0	0
7_x	0	2	2	0	0	2	2	0	0	2	0	2	2	0	2	0
8_x	0	4	0	4	0	4	0	4	0	0	0	0	0	0	0	0
9_x	0	2	0	2	0	2	0	2	4	0	4	0	0	0	0	0
A_x	0	0	0	0	4	4	0	0	0	2	2	0	2	0	0	2
B_x	0	0	2	2	0	0	2	2	0	2	0	2	2	0	2	0
C_x	0	0	0	0	0	0	0	0	4	2	0	2	0	2	4	2
D_x	0	2	0	2	0	2	0	2	0	2	0	2	0	2	0	2
E_x	0	4	0	0	4	0	0	0	0	0	2	2	2	2	0	0
F_x	0	0	2	2	0	0	2	2	0	0	0	0	2	2	2	2

4.2. Undisturbed Bits of S-Box and inverse S-Box

Undisturbed Bits are a technique to find the exact output bit difference of S-Box with probability one. Since S-Box is a non-linear operation and its output difference can be found with some probability by using DDT, the undisturbed bits give the exact output difference bits of S-Box with probability one. Undisturbed bits help us find longer impossible differentials. It was used for the first time in (Tezcan, 2014). Also, they are used in (Tezcan, 2016) and (Tezcan, 2020) to find the undisturbed bits of the S-Box of ASCON (Dobraunig, Mendel, et al., 2019).

For example, when the input difference of S-Box is 8 (1000), the output difference of S-Box can be 4 (01**0**0), 5 (01**0**1), 9 (10**0**1) and C (11**0**0). It is obvious that the first bit of output difference remains invariant. The other bits can be "1" or "0"; therefore, they are denoted as "?". To conclude, if the input difference of S-Box is (1000), the output difference of S-Box will be (??0?) with probability one. The undisturbed bits will be used in Section 4.3.2. Table 4.4 shows the Undisturbed Bits of Shadow-512's S-Box and inverse S-Box.

Table 4.4: Undisturbed Bits of S-Box and inverse S-Box

Undisturbed Bits of S-Box	
Input Difference	Output Difference
0010	???1
0100	??10
0110	???1
1000	??0?
1100	??1?
?000	??0?
?100	??1?
0?10	???1

Undisturbed Bits of inverse S-Box	
Input Difference	Output Difference
0100	1???
0101	???0
1000	0??1
1100	1???
1101	???1
?100	1???

4.3. 6-Step Impossible Differential Distinguisher of Shadow-512

In this section, two different 6-Step impossible differential distinguishers of Shadow-512 will be explained. We found 6-Step impossible differential distinguishers by using the 5-Step truncated differential, which was introduced in (Derbez et al., 2020). The detailed explanation about 5-Step truncated differential was given in Section 3.2. Firstly, one more step is added to the top of 5-Step truncated differential in the forward direction to obtain 6-Step impossible differential distinguisher. The impossible differential distinguisher is obtained between Step-0 and Step-1. Secondly, one step and an inverse D-Box operation are added to the bottom of the 5-Step truncated differential in the backward direction to obtain another 6-Step impossible differential distinguisher between Step-4 and Step-5.

4.3.1. Adding One More Step in the Forward Direction

In this section, one of our 6-Step impossible differential distinguisher will be explained. The 6-Step impossible differential distinguisher has two parts. Firstly usage of the 5-Step truncated differential of (Derbez et al., 2020) will be summarized and then one step that is added to the top of the 5-Step truncated differential will be shown. The 5-Step truncated differential starts from Step-3 and consists of two steps in the forward direction and two steps in the backward direction. The one step differential starts from Step-0 in the forward direction. The idea is to set the Bundle-3 difference to a nonzero value at the end of Step-0. Thus, the difference of Bundle-3 of Step-0 does not match the Bundle-3 difference of Step-1. The impossible differential will be obtained between the Step-0 output difference and Step-1 input difference. Fig. 4.1 shows the 6-Step impossible differential distinguisher.

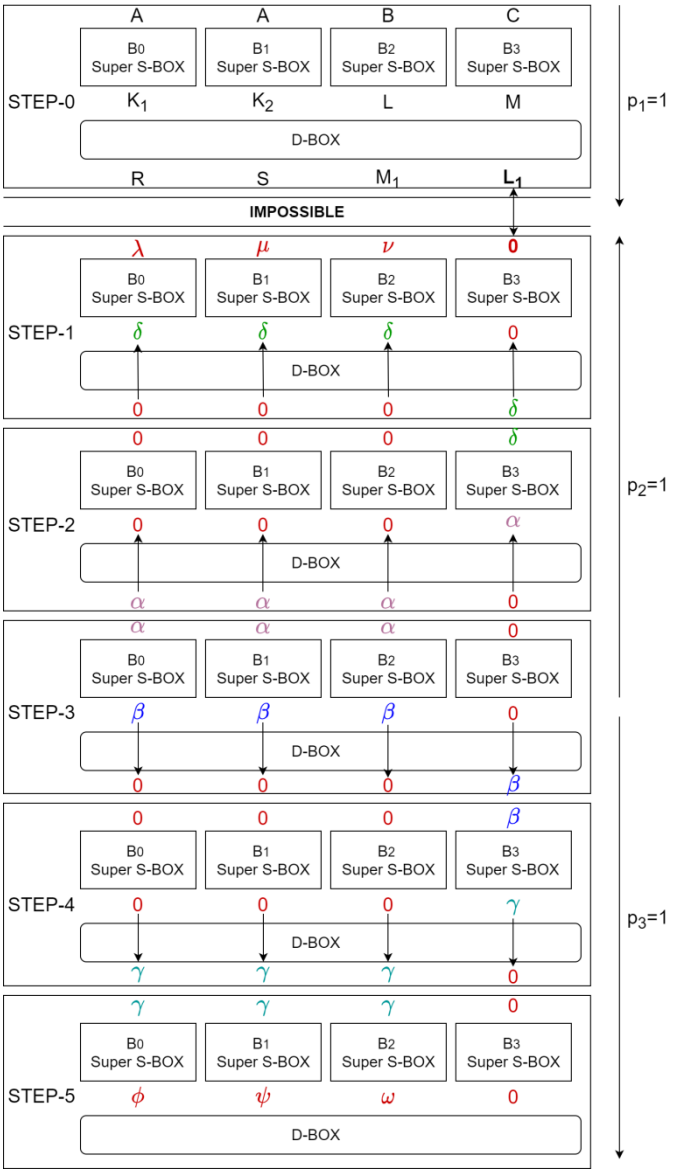

Figure 4.1: 6-Step Impossible Differential Distinguisher of Shadow-512 that starts from Step-3

Algorithm-3 details the 6-Step impossible differential distinguisher. *Let σ_j denote the Super S-Box operation and j denotes the bundle index.*

Algorithm − 3 6 − Step distinguisher by adding one step to the top

Impossible: The difference of Step-1, which is (R, S, M_1, L_1) does not match the difference of Step-1, which is $(\lambda, \mu, \nu, 0)$ in the middle.

i. Choose a random pair such that $y \in \mathbb{F}_2^{128}$ and $(y \oplus \beta) \in \mathbb{F}_2^{128}$ in Step-3. β must be set to zero on the 0th, 1st,2nd,3rd columns of a bundle.

ii. Choose a random state $z \in \mathbb{F}_2^{128}$.

iii. Compute $x = \sigma_j^{-1}(y)$ and $x + \alpha = \sigma_j^{-1}(y + \beta)$ for $0 \leq j \leq 2$. Set the states at Step-3 such that

$$X_3 = (x, x, x, z) \; and \; X_3' = (x + \alpha, x + \alpha, x + \alpha, z) \rightarrow \Delta = (\alpha, \alpha, \alpha, 0)$$

iv. Iterate Step-4 and Step-5 on X_3 and X_3' to obtain $(\phi, \psi, \omega, 0)$ in Step-5

v. The difference of Step-5 $(\phi, \psi, \omega, 0)$ cannot come from the input difference (A, A, B, C) in Step-0 if the difference of Step-3 is $(\beta, \beta, \beta, 0)$.

According to Algorithm-3, the truncated trail starts from Step-3 with the difference $(\alpha, \alpha, \alpha, 0)$. The first three bundles are in a 3-Identical state and their differences are the same. After two steps in the forward direction, the difference will become $(\phi, \psi, \omega, 0)$ with probability one. The difference value of the first three bundles can be any value, but the difference of Bundle-3 will be definitely zero in Step-5. In a similar way, two steps in the backward direction are applied to bundles and the difference $(\alpha, \alpha, \alpha, 0)$ will become $(\lambda, \mu, \nu, 0)$. The difference value of Bundle-0, Bundle-1 and Bundle-2 can be any value other than zero; however, the difference of Bundle-3 will be absolutely zero at the beginning of Step-1. This is 5-Step truncated differential of (Derbez et al., 2020) and the detailed explanation was given in Section 3.2.

In Table 4.5, the one step truncated differential that starts from Step-0 is shown. The purpose is to prove the nonzero Bundle-3 difference after one step in the forward direction.

Table 4.5: 1-Step in the forward direction to obtain 6-Step impossible differential distinguisher

	Bundle-0	**Bundle-1**	**Bundle-2**	**Bundle-3**
Initialization	$P_{0,1} \oplus P_{0,2} = \mathrm{A}$	$P_{1,1} \oplus P_{1,2} = \mathrm{A}$	$P_{2,1} \oplus P_{2,2} = \mathrm{B}$	$P_{3,1} \oplus P_{3,2} = \mathrm{C}$
S-Box	$P'_{0,1} \oplus P'_{0,2} = D$	$P'_{1,1} \oplus P'_{1,2} = D$	$P'_{2,1} \oplus P'_{2,2} = F$	$P'_{3,1} \oplus P'_{3,2} = G$
L-Box	$P''_{0,1} \oplus P''_{0,2} = E$	$P''_{1,1} \oplus P''_{1,2} = E$	$P''_{2,1} \oplus P''_{2,2} = H$	$P''_{3,1} \oplus P''_{3,2} = J$
RC	$P'''_{0,1} \oplus P'''_{0,2} = E$	$P'''_{1,1} \oplus P'''_{1,2} = E$	$P'''_{2,1} \oplus P'''_{2,2} = H$	$P'''_{3,1} \oplus P'''_{3,2} = J$
S-Box	$P''''_{0,1} \oplus P''''_{0,2} = K_1$	$P''''_{1,1} \oplus P''''_{1,2} = K_2$	$P''''_{2,1} \oplus P''''_{2,2} = L$	$P''''_{3,1} \oplus P''''_{3,2} = M$
D-Box	$(P''''_{1,1} \oplus P''''_{2,1} \oplus P''''_{3,1}) \oplus (P''''_{1,2} \oplus P''''_{2,2} \oplus P''''_{3,2})$ $= K_2 \oplus L \oplus M$ $= R$	$(P''''_{0,1} \oplus P''''_{2,1} \oplus P''''_{3,1}) \oplus (P''''_{0,2} \oplus P''''_{2,2} \oplus P''''_{3,2})$ $= K_1 \oplus L \oplus M$ $= S$	$(P''''_{0,1} \oplus P''''_{1,1} \oplus P''''_{3,1}) \oplus (P''''_{0,2} \oplus P''''_{1,2} \oplus P''''_{3,2})$ $= K_1 \oplus K_2 \oplus M$ $= M_1$	$(P''''_{0,1} \oplus P''''_{1,1} \oplus P''''_{2,1}) \oplus (P''''_{0,2} \oplus P''''_{1,2} \oplus P''''_{2,2})$ $= K_1 \oplus K_2 \oplus L$ $= \boldsymbol{L_1}$

To obtain nonzero Bundle-3 difference at the end of Step-0, there are some constraints about defining Step-0 pairs in the Initialization part:

- The value of difference "A" can be any value. The only condition is that Bundle-0 and Bundle-1 pairs should be chosen the same. $P_{0,1} = P_{1,1}$ and $P_{0,2} = P_{1,2}$ should be satisfied.
- The value of the difference "B" must be different from approximately 2^{76} different difference values that will be explained later.
- The value of the difference "C" can be any value. Input pairs that are chosen for Bundle-3 can be random.

S-Box and L-Box Layer:

In the first S-Box layer, since Bundle-0 and Bundle-1 pairs are the same, the output of the S-Box layer and their differences will become the same value. Moreover, it is obvious that after the L-Box layer, Bundle-0 and Bundle-1 pairs and their differences are still the same since inputs of the L-Box layer are the same.

The input pairs are identical for Bundle-0 and Bundle-1.

$P_{0,1} = P_{1,1}$ *and* $P_{0,2} = P_{1,2}$

- For Bundle-0:

$SBOX(P_{0,1}) = P'_{0,1}$, $SBOX(P_{0,2}) = P'_{0,2}$

$\Delta_0^{sbox-1}(P'_{0,1}, P'_{0,2}) = P'_{0,1} \oplus P'_{0,2} = D$

$LBOX(P'_{0,1}) = P''_{0,1}$, $LBOX(P'_{0,2}) = P''_{0,2}$

$\Delta_0^{lbox}(P''_{0,1}, P''_{0,2}) = P''_{0,1} \oplus P''_{0,2} = E$

- For Bundle-1:

$SBOX(P_{1,1}) = P'_{1,1}$, $SBOX(P_{1,2}) = P'_{1,2}$

$\Delta_1^{sbox-1}(P'_{1,1}, P'_{1,2}) = P'_{1,1} \oplus P'_{1,2} = D$

$LBOX(P'_{1,1}) = P''_{1,1}$, $LBOX(P'_{1,2}) = P''_{1,2}$

$\Delta_1^{lbox}(P''_{1,1}, P''_{1,2}) = P''_{1,1} \oplus P''_{1,2} = E$

Output pairs will be identical for Bundle-0 and Bundle-1. $P''_{0,1} = P''_{1,1}$ and $P''_{0,2} = P''_{1,2}$.

- For Bundle-2:

$SBOX(P_{2,1}) = P'_{2,1}$, $SBOX(P_{2,2}) = P'_{2,2}$

$\Delta_2^{sbox-1}(P'_{2,1}, P'_{2,2}) = P'_{2,1} \oplus P'_{2,2} = F$

$LBOX(P'_{2,1}) = P''_{2,1}$, $LBOX(P'_{2,2}) = P''_{2,2}$

$\Delta_2^{lbox}(P''_{2,1}, P''_{2,2}) = P''_{2,1} \oplus P''_{2,2} = H$

- For Bundle-3:

$SBOX(P_{3,1}) = P'_{3,1}$, $SBOX(P_{3,2}) = P'_{3,2}$

$\Delta_3^{sbox-1}(P'_{3,1}, P'_{3,2}) = P'_{3,1} \oplus P'_{3,2} = G$

$LBOX(P'_{3,1}) = P''_{3,1}$, $LBOX(P'_{3,2}) = P''_{3,2}$

$\Delta_3^{lbox}(P''_{3,1}, P''_{3,2}) = P''_{3,1} \oplus P''_{3,2} = J$

Round Constant Addition:

The only part that changes the value of Bundle-0 and Bundle-1 pairs is the round constant addition part. As explained in Section 2.4.3, round constant operation is applied to the different columns for Bundle-0 and Bundle-1. The round constant is added to 0^{th} column for Bundle-0 and 1^{st} column of Bundle-1. After round constant

addition part, Bundle-0 and Bundle-1 pairs are slightly different from each other; however, their differences are still the same since round constant addition is just an XOR operation. Although it affects the values of pairs, the differences are not affected by round constant.

- For Bundle-0:

$RC(P''_{0,1}) = P'''_{0,1}$, $\quad RC(P''_{0,2}) = P'''_{0,2}$

$\Delta_0^{rc}(P'''_{0,1}, P'''_{0,2}) = P'''_{0,1} \oplus P'''_{0,2} = E$

- For Bundle-1:

$RC(P''_{1,1}) = P'''_{1,1}$, $\quad RC(P''_{1,2}) = P'''_{1,2}$

$\Delta_1^{rc}(P'''_{1,1}, P'''_{1,2}) = P'''_{1,1} \oplus P'''_{1,2} = E$

The output pairs are slightly different from each other. $P'''_{0,1} \approx P'''_{1,1}$ and $P'''_{0,2} \approx P'''_{1,2}$.

In Table 4.6, the bits that are represented by grey are different for Bundle-0 and Bundle-1 input pairs. All the remaining bits are the same for Bundle-0 and Bundle-1 pairs.

Table 4.6: The bits that are different for Bundle-0 and Bundle-1

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**

- For Bundle-2:

$RC(P''_{2,1}) = P'''_{2,1}$, $\quad RC(P''_{2,2}) = P'''_{2,2}$

$\Delta_2^{rc}(P'''_{2,1}, P'''_{2,2}) = P'''_{2,1} \oplus P'''_{2,2} = H$

- For Bundle-3:

$RC(P''_{3,1}) = P'''_{3,1}$, $\quad RC(P''_{3,2}) = P'''_{3,2}$

$\Delta_1^{rc}(P'''_{3,1}, P'''_{3,2}) = P'''_{3,1} \oplus P'''_{3,2} = J$

Second S-Box Layer:

The operation continues with the second S-Box layer. After the second S-Box layer, Bundle-0 and Bundle-1 pairs and their differences are still the same except for the 0^{th} and 1^{st} columns since the same input differences may lead to different output differences if two input columns of S-Box are different from each other. In this case, only two columns of Bundle-0 and Bundle-1 may be different and the other columns are definitely the same. Since there is a slight possibility that 0^{th} and 1^{st} columns are different for Bundle-0 and Bundle-1 pairs, it is assumed that they are different.

- For Bundle-0:

$$SBOX(P'''_{0,1}) = P''''_{0,1}, \qquad SBOX(P'''_{0,2}) = P''''_{0,2}$$

$$\Delta_0^{sbox-2}(P''''_{0,1}, P''''_{0,2}) = P''''_{0,1} \oplus P''''_{0,2} = K_1$$

- For Bundle-1:

$$SBOX(P'''_{1,1}) = P''''_{1,1}, \qquad SBOX(P'''_{1,2}) = P''''_{1,2}$$

$$\Delta_1^{sbox-2}(P''''_{1,1}, P''''_{1,2}) = P''''_{1,1} \oplus P''''_{1,2} = K_2$$

The output pairs of Bundle-0 and Bundle-1 are almost the same except for the least significant two columns. $P''''_{0,1} \approx P''''_{1,1}$ and $P''''_{0,2} \approx P''''_{1,2}$. Therefore, $K_1 \approx K_2$.

- For Bundle-2:

$$SBOX(P'''_{2,1}) = P''''_{2,1}, \qquad SBOX(P'''_{2,2}) = P''''_{2,2}$$

$$\Delta_2^{sbox-2}(P''''_{2,1}, P''''_{2,2}) = P''''_{2,1} \oplus P''''_{2,2} = L$$

- For Bundle-3:

$$SBOX(P'''_{3,1}) = P''''_{3,1}, \qquad SBOX(P'''_{3,2}) = P''''_{3,2}$$

$$\Delta_3^{sbox-2}(P''''_{3,1}, P''''_{3,2}) = P''''_{3,1} \oplus P''''_{3,2} = M$$

D-Box Layer:

In the D-Box layer, as mentioned earlier in Section 2.4.4, three of four bundles are XORed with each other to construct other bundle.

- For Bundle-0:

$$DBOX(P''''_{1,1}, P''''_{2,1}, P''''_{3,1}) = P''''_{1,1} \oplus P''''_{2,1} \oplus P''''_{3,1} = P'''''_{0,1}$$

$$DBOX(P''''_{1,2}, P''''_{2,2}, P''''_{3,2}) = P''''_{1,2} \oplus P''''_{2,2} \oplus P''''_{3,2} = P'''''_{0,2}$$

$$\Delta_0^{dbox} = DBOX\left(P_{1,1}'''', P_{2,1}'''', P_{3,1}''''\right) \oplus DBOX\left(P_{1,2}'''', P_{2,2}'''', P_{3,2}''''\right)$$

$$\Delta_0^{dbox} = P_{0,1}''''' \oplus P_{0,2}''''' = R$$

- For Bundle-1:

$$DBOX\left(P_{0,1}'''', P_{2,1}'''', P_{3,1}''''\right) = P_{0,1}'''' \oplus P_{2,1}'''' \oplus P_{3,1}'''' = P_{1,1}'''''$$

$$DBOX\left(P_{0,2}'''', P_{2,2}'''', P_{3,2}''''\right) = P_{0,2}'''' \oplus P_{2,2}'''' \oplus P_{3,2}'''' = P_{1,2}'''''$$

$$\Delta_1^{dbox} = DBOX\left(P_{0,1}'''', P_{2,1}'''', P_{3,1}''''\right) \oplus DBOX\left(P_{0,2}'''', P_{2,2}'''', P_{3,2}''''\right)$$

$$\Delta_1^{dbox} = P_{1,1}''''' \oplus P_{1,2}''''' = S$$

It is known that Bundle-0 and Bundle-1 pairs are almost the same before the D-Box operation. Therefore, the difference value of Bundle-2 will come from Bundle-3 and the difference value of Bundle-3 will come from Bundle-2 after the D-Box layer.

- For Bundle-2:

$$DBOX\left(P_{0,1}'''', P_{1,1}'''', P_{3,1}''''\right) = P_{0,1}'''' \oplus P_{1,1}'''' \oplus P_{3,1}'''' = P_{2,1}'''''$$

$$DBOX\left(P_{0,2}'''', P_{1,2}'''', P_{3,2}''''\right) = P_{0,2}'''' \oplus P_{1,2}'''' \oplus P_{3,2}'''' = P_{2,2}'''''$$

Recall that $P_{0,1}'''' \approx P_{1,1}''''$ and $P_{0,2}'''' \approx P_{1,2}''''$; therefore, $P_{2,1}''''' \approx P_{3,1}''''$ and $P_{2,2}''''' \approx P_{3,2}''''$.

$$\Delta_2^{dbox} = DBOX\left(P_{0,1}'''', P_{1,1}'''', P_{3,1}''''\right) \oplus DBOX\left(P_{0,2}'''', P_{1,2}'''', P_{3,2}''''\right) \approx P_{3,1}'''' \oplus P_{3,2}'''' = M_1$$

It can be said that $M_1 \approx M$.

- For Bundle-3:

$$DBOX\left(P_{0,1}'''', P_{1,1}'''', P_{2,1}''''\right) = P_{0,1}'''' \oplus P_{1,1}'''' \oplus P_{2,1}'''' = P_{3,1}'''''$$

$$DBOX\left(P_{0,2}'''', P_{1,2}'''', P_{2,2}''''\right) = P_{0,2}'''' \oplus P_{1,2}'''' \oplus P_{2,2}'''' = P_{3,2}'''''$$

Recall that $P_{0,1}'''' \approx P_{1,1}''''$ and $P_{0,2}'''' \approx P_{1,2}''''$; therefore, $P_{3,1}''''' \approx P_{2,1}''''$ and $P_{3,2}''''' \approx P_{2,2}''''$.

$$\Delta_3^{dbox} = DBOX\left(P_{0,1}'''', P_{1,1}'''', P_{2,1}''''\right) \oplus DBOX\left(P_{0,2}'''', P_{1,2}'''', P_{2,2}''''\right) \approx P_{2,1}'''' \oplus P_{2,2}'''' = L_1$$

It can be said that $L_1 \approx L$.

After the D-Box layer, the difference value of Bundle-3 comes from $P_{2,1}'''' \oplus P_{2,2}''''$ except for the 0th and 1st column. The idea is that if $P_{2,1}'''' \oplus P_{2,2}'''' = L_1$ is not equal to zero, Bundle-3 cannot have zero difference after the D-Box layer of Step-0. This brings up the question. Which input difference should not be given to the Bundle-2 at the beginning of Step-0 so that the difference value of Bundle-2 ($P_{2,1}'''' \oplus P_{2,2}''''$) after the second S-Box layer is not equal to zero? Thus, it will be proven that the value of "L_1" cannot be zero.

In Table 4.7, the difference value of Bundle-2 that should not be obtained after the second S-Box layer is shown. The difference value of the least significant two columns of Bundle-2 can be any value. Since Bundle-0, Bundle-1 and Bundle-2 are XORed with each other to build Bundle-3 in the D-Box layer, the difference value of the least significant two columns of Bundle-3 can be zero. Therefore, if it is guaranteed that the difference of other columns of Bundle-2 cannot be zero, the difference of Bundle-3 will be nonzero after the D-Box.

Table 4.7: The difference of Bundle-2 that should not be obtained after the second S-Box layer

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Now, we should think in the opposite way. If Table 4.7 represents the output difference of Bundle-2 after the second S-Box layer, the output difference of L-Box must also be identical to Table 4.7. Because the only way that makes the output difference zero after the S-Box operation is setting the input difference to zero. Therefore, in Table 5.8, it is shown that there are 36 possible Bundle-2 difference values that give the zero output difference except for the 0^{th} and 1^{st} column after the L-Box layer. Each cell represents a difference value of a Bundle-2 that should not be given as input difference of L-Box.

Let $\Delta^{sbox-2} = \begin{pmatrix} 0000000? \\ 0000000? \\ 0000000? \\ 0000000? \end{pmatrix}$ and then $Inv\text{-}\ SBOX \begin{pmatrix} 0000000? \\ 0000000? \\ 0000000? \\ 0000000? \end{pmatrix} = \begin{pmatrix} 0000000? \\ 0000000? \\ 0000000? \\ 0000000? \end{pmatrix}$

Therefore, $Inv\text{-}\ LBOX \begin{pmatrix} 0000000? \\ 0000000? \\ 0000000? \\ 0000000? \end{pmatrix} = \begin{pmatrix} 6a1b93b3 \\ 600ed073 \\ 6a1b93b3 \\ 600ed073 \end{pmatrix}$

Table 4.8: Possible input differences of L-Box that makes the output difference zero

0x6a1b93b3 0x600ed073 0x6a1b93b3 0x600ed073	0x6a1b93b3 0x600ed073 0xbf188a19 0x65047193	0x6a1b93b3 0x600ed073 0x600ed073 0xb50dc9d9	0x6a1b93b3 0x600ed073 0xb50dc9d9 0xb0076839	0x6a1b93b3 0x600ed073 0x060cc140 0xb3010899	0x6a1b93b3 0x600ed073 0xb3010899 0x030660a0
0xbf188a19 0x65047193 0xbf188a19 0x65047193	0xbf188a19 0x65047193 0x600ed073 0xb50dc9d9	0xbf188a19 0x65047193 0xb50dc9d9 0xb0076839	0xbf188a19 0x65047193 0x060cc140 0xb3010899	0xbf188a19 0x65047193 0xb3010899 0x030660a0	0xbf188a19 0x65047193 0x6a1b93b3 0x600ed073
0x600ed073 0xb50dc9d9 0x600ed073 0xb50dc9d9	0x600ed073 0xb50dc9d9 0xb50dc9d9 0xb0076839	0x600ed073 0xb50dc9d9 0x060cc140 0xb3010899	0x600ed073 0xb50dc9d9 0xb3010899 0x030660a0	0x600ed073 0xb50dc9d9 0x6a1b93b3 0x600ed073	0x600ed073 0xb50dc9d9 0xbf188a19 0x65047193
0xb50dc9d9 0xb0076839 0xb50dc9d9 0xb0076839	0xb50dc9d9 0xb0076839 0x060cc140 0xb3010899	0xb50dc9d9 0xb0076839 0xb3010899 0x030660a0	0xb50dc9d9 0xb0076839 0x6a1b93b3 0x600ed073	0xb50dc9d9 0xb0076839 0xbf188a19 0x65047193	0xb50dc9d9 0xb0076839 0x600ed073 0xb50dc9d9
0x60cc140 0xb3010899 0x060cc140 0xb3010899	0x060cc140 0xb3010899 0xb3010899 0x030660a0	0x060cc140 0xb3010899 0x6a1b93b3 0x600ed073	0x060cc140 0xb3010899 0xbf188a19 0x65047193	0x060cc140 0xb3010899 0x600ed073 0xb50dc9d9	0x060cc140 0xb3010899 0xb50dc9d9 0xb0076839
0xb3010899 0x030660a0 0xb3010899 0x030660a0	0xb3010899 0x030660a0 0x6a1b93b3 0x600ed073	0xb3010899 0x030660a0 0xbf188a19 0x65047193	0xb3010899 0x030660a0 0x600ed073 0xb50dc9d9	0xb3010899 0x030660a0 0xb50dc9d9 0xb0076839	0xb3010899 0x030660a0 0x060cc140 0xb3010899

The difference values that are shown in Table 4.8 also can be considered as the output difference of the first S-Box layer. According to DDT, all possible input difference values of the first S-Box can be found practically.

For example, Bundle-2 output difference after the first S-Box layer is $\begin{pmatrix} 6a1b93b3 \\ 600ed073 \\ 6a1b93b3 \\ 600ed073 \end{pmatrix}$.

Binary representation of this difference is: $\begin{pmatrix} 01101010000110111001001110110011 \\ 01100000000011101101000001110011 \\ 01101010000110111001001110110011 \\ 01100000000011101101000001110011 \end{pmatrix}$.

Recall that S-Box operation is applied column by column and this difference value is the output difference of the first S-Box layer. With the help of DDT, all possible input differences of S-Boxes can be found.

0 1 1 0 1 0 1 0 0 0 0 1 1 0 1 1 1 0 0 1 0 0 1 1 1 0 1 1 0 0 1 1
0 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 0 0 0 0 0 1 1 1 0 0 1 1
0 1 1 0 1 0 1 0 0 0 0 1 1 0 1 1 1 0 0 1 0 0 1 1 1 0 1 1 0 0 1 1
0 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 0 0 0 0 0 1 1 1 0 0 1 1
↓ ↓
1 8 8 1 6 1 6 1 1 1 1 6 8 4 8 6 8 4 1 8 1 1 6 6 6 4 8 8 1 1 8 8 → *The number of possible input differences for each column.*

If we multiply the numbers of all possible input differences for each column, it is seen that there are $2^{54.0947}$ possible input differences of S-Box for $\begin{pmatrix} 6a1b93b3 \\ 600ed073 \\ 6a1b93b3 \\ 600ed073 \end{pmatrix}$. Recall that, there are 36 different output differences that should not be obtained after the first S-Box operation. Therefore, when this calculation is made for all 36 different output difference structures, it will be seen that there are approximately 2^{76} input difference values that cannot be given as input difference of the first S-Box. To conclude, at the beginning of Step-0, there are approximately ($2^{128} - 2^{76}$) possible input differences that can be given to Bundle-2.

By this point, it was proven that Bundle-0 and Bundle-1 almost have identical difference values and identical pairs except 0th and 1st column after the second S-Box layer. Moreover, it is guaranteed that the difference value of Bundle-2 cannot be zero after the second S-Box layer. Finally, in the D-Box layer, since Bundle-0, Bundle-1 and Bundle-2 are XORed with each other to construct Bundle-3, the difference of Bundle-3 cannot be zero with probability one. The difference (R, S, M_1, L_1)cannot match the difference $(\lambda, \mu, \nu, 0)$ between Step-0 and Step-1. Therefore, it can be said that the output difference of Step-5, which is $(\phi, \psi, \omega, 0)$ cannot come from the input difference of Step-0, which is (A, A, B, C) with probability one if the output difference of Super S-Box of Step-3 is $(\beta, \beta, \beta, 0)$.

$$\Delta_o = (\phi, \psi, \omega, 0) \nleftarrow \Delta_i = (A, A, B, C) \; with \; probability \; 1$$

To sum up, 6-Step impossible differential distinguisher was obtained. Since at the beginning of Step-0, the input difference is defined as (A, A, B, C), there are $2^{256} \times (2^{128} - 2^{72})$ possible input difference structures that can be generated. It can be said that the number of possible inputs is approximately 2^{384}. Therefore, this impossible differential holds with the probability of 2^{-128} for a random permutation and due to the birthday paradox, the impossible differential distinguisher needs 2^{64} different pairs to distinguish Shadow-512 from a random permutation with probability one. According to (Derbez et al., 2020), the probability of their 6-Step truncated differential distinguisher is $2^{-16.245}$. Compared to their result, our distinguisher works with probability one. Although our distinguisher has a better probability, a distinguishing attack on a random permutation would require around 2^{64} pairs due to the filtering conditions on the input pairs.

4.3.2. Adding One More Step in the Backward Direction

In this section, another 6-Step impossible differential distinguisher will be explained. The 6-Step impossible differential distinguisher consists of two parts. Firstly, 5-Step truncated differential, which was explained in Section 3.2 will be summarized. Secondly, the one step in the backward direction that is added to the bottom of the 5-Step truncated differential will be explained. The 5-Step truncated differential starts from Step-2 and the difference propagates with probability one along with two steps in the forward direction and two steps in the backward direction. The one step starts from Step-5 and finishes up with inverse D-Box operation of Step-4 with probability one. The idea is to set the Bundle-3 difference to a nonzero value after one step and inverse D-Box operation with probability one so that it does not match the difference of Bundle-3, which is zero in Step-4.

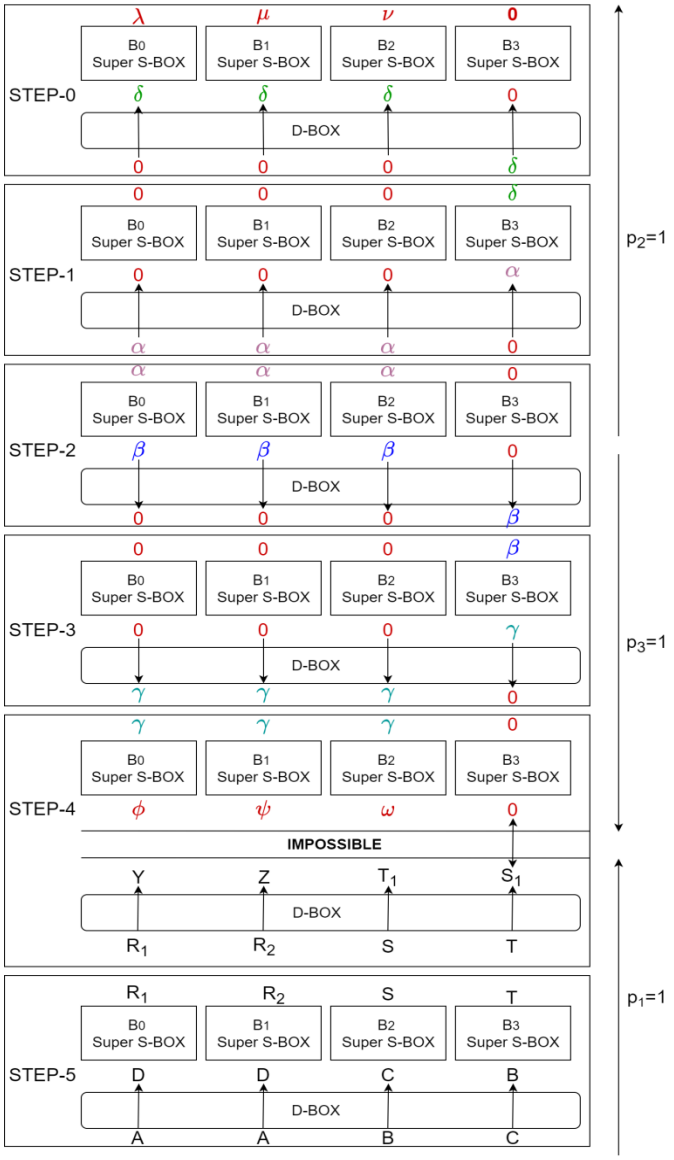

Figure 4.2: 6-Step Impossible Differential Distinguisher of Shadow-512 that starts from Step-2

Algorithm-4 details the 6-Step impossible differential distinguisher. *Let σ_j denote the Super S-Box operation and j denotes the bundle index.*

Algorithm 4 $6-$Step distinguisher by adding one step to the bottom

Impossible: The difference of Step-4, which is $(\phi, \psi, \omega, 0)$ does not match the difference $(0, z, z, z)$ in the middle.

i. Choose a random pair such that $y \in \mathbb{F}_2^{128}$ and $(y \oplus \beta) \in \mathbb{F}_2^{128}$ in Step-3. β must be set to zero on the 0^{th}, 1^{st},2^{nd},3^{rd} columns of a bundle.

ii. Choose a random $z \in \mathbb{F}_2^{128}$.

iii. Compute $x = \sigma_j^{-1}(y)$ and $x + \alpha = \sigma_j^{-1}(y + \beta)$ for $0 \leq j \leq 2$. Set the states at Step-2 such that

$$X_2 = (x, x, x, z) \; and \; X_2' = (x + \alpha, x + \alpha, x + \alpha, z) \rightarrow \Delta = (\alpha, \alpha, \alpha, 0)$$

iv. Invert Step-1 and Step-0 to obtain $(\lambda, \mu, \nu, 0)$ at the beginning of Step-0.

v. The difference of Step-5 (A, A, B, C) cannot come from the input difference $(\lambda, \mu, \nu, 0)$ in Step-0 if the difference of Step-2 is $(\beta, \beta, \beta, 0)$.

According to Algorithm-4, the truncated trail starts from Step-2 with the difference $(\alpha, \alpha, \alpha, 0)$. The first three bundles are in a 3-Identical state. After two steps in the backward direction, the difference will become $(\lambda, \mu, \nu, 0)$ with probability one. Similarly, two steps in the forward direction are applied to the bundles in Step-2 and the difference $(\alpha, \alpha, \alpha, 0)$ will become $(\phi, \psi, \omega, 0)$ after Super S-Box operation of Step-4 with probability one. This is the 5-Step truncated differential of (Derbez et al., 2020) and the detailed explanation of this was given in Section 3.2. As can be seen in Fig. 4.2, the difference of Bundle-3 is zero after the Super S-Box of Step-4.

In Table 4.9, the one step that starts from Step-0 is shown. The purpose is adding one step and an inverse D-Box to obtain the nonzero Bundle-3 difference at the end of the inverse D-Box of Step-4.

Table 4.9: 1-Step in the backward direction to obtain 6-Step impossible differential distinguisher

	Bundle-0	**Bundle-1**	**Bundle-2**	**Bundle-3**
Initialization	$P_{0,1} \oplus P_{0,2} = \mathrm{A}$	$P_{1,1} \oplus P_{1,2} = \mathrm{A}$	$P_{2,1} \oplus P_{2,2} = \mathrm{B}$	$P_{3,1} \oplus P_{3,2} = \mathrm{C}$
Inverse D-Box	$(P_{1,1} \oplus P_{2,1} \oplus P_{3,1}) \oplus (P_{1,2} \oplus P_{2,2} \oplus P_{3,2})$ $\mathrm{A} \oplus B \oplus C = D$	$(P_{0,1} \oplus P_{2,1} \oplus P_{3,1}) \oplus (P_{0,2} \oplus P_{2,2} \oplus P_{3,2})$ $\mathrm{A} \oplus B \oplus C = D$	$(P_{0,1} \oplus P_{1,1} \oplus P_{3,1}) \oplus (P_{0,2} \oplus P_{1,2} \oplus P_{3,2})$ $\mathrm{A} \oplus A \oplus C = C$	$(P_{0,1} \oplus P_{1,1} \oplus P_{2,1}) \oplus (P_{0,2} \oplus P_{1,2} \oplus P_{2,2})$ $\mathrm{A} \oplus A \oplus B = B$
Inverse S-Box	$P'_{0,1} \oplus P'_{0,2} = E$	$P'_{1,1} \oplus P'_{1,2} = E$	$P'_{2,1} \oplus P'_{2,2} = H$	$P'_{3,1} \oplus P'_{3,2} = J$
RC	$P''_{0,1} \oplus P''_{0,2} = E$	$P''_{1,1} \oplus P''_{1,2} = E$	$P''_{2,1} \oplus P''_{2,2} = H$	$P''_{3,1} \oplus P''_{3,2} = J$
Inverse L-Box	$P'''_{0,1} \oplus P'''_{0,2} = K$	$P'''_{1,1} \oplus P'''_{1,2} = K$	$P'''_{2,1} \oplus P'''_{2,2} = L$	$P'''_{3,1} \oplus P'''_{3,2} = M$
Inverse S-Box	$P''''_{0,1} \oplus P''''_{0,2} = R_1$	$P''''_{1,1} \oplus P''''_{1,2} = R_2$	$P''''_{2,1} \oplus P''''_{2,2} = S$	$P''''_{3,1} \oplus P''''_{3,2} = T$
Inverse D-Box	$(P''''_{1,1} \oplus P''''_{2,1} \oplus P''''_{3,1}) \oplus (P''''_{1,2} \oplus P''''_{2,2} \oplus P''''_{3,2})$ $= R_2 \oplus S \oplus T$ $= Y$	$(P''''_{0,1} \oplus P''''_{2,1} \oplus P''''_{3,1}) \oplus (P''''_{0,2} \oplus P''''_{2,2} \oplus P''''_{3,2})$ $= R_1 \oplus S \oplus T$ $= Z$	$(P''''_{0,1} \oplus P''''_{1,1} \oplus P''''_{3,1}) \oplus (P''''_{0,2} \oplus P''''_{1,2} \oplus P''''_{3,2})$ $= R_1 \oplus R_2 \oplus T$ $= T_1$	$(P''''_{0,1} \oplus P''''_{1,1} \oplus P''''_{2,1}) \oplus (P''''_{0,2} \oplus P''''_{1,2} \oplus P''''_{2,2})$ $= R_1 \oplus R_2 \oplus S$ $= S_1$

Note that before the D-Box layer of Step-5, there is a round constant addition part. Since round constant addition does not affect the difference value and also it is known which round constant value is added to bundles, this round constant addition part is skipped. Therefore, the pairs should be defined after round constant addition part of Step-5.

To obtain nonzero Bundle-3 difference at the end of Super S-Box of Step-4, there are some constraints about defining Step-5 pairs in the Initialization part:

- The value of difference "A" can be any value. The only condition is that Bundle-0 and Bundle-1 pairs are chosen the same. $P_{0,1} = P_{1,1}$ and $P_{0,2} = P_{1,2}$ should be satisfied.
- The value of the difference "B" can be any value.
- The value of the difference "C" has an exact value. One bit difference should be given to the pairs. The difference value can be seen in Table 4.10.

In this iteration, each row of a bundle is represented with binary notation. The bits that are shown as "?" can be 1 or 0.

Table 4.10: Initial values of bundles at the end of Step-5

	Initialization
Bundle-0	???????????????????????????????? ???????????????????????????????? ???????????????????????????????? ???????????????????????????????? Value is A
Bundle-1	???????????????????????????????? ???????????????????????????????? ???????????????????????????????? ???????????????????????????????? Value is A
Bundle-2	???????????????????????????????? ???????????????????????????????? ???????????????????????????????? ???????????????????????????????? Value is B
Bundle-3	00001000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 Value is C

Firstly, the difference value is given to the pairs according to constraints. Then, the distribution of the difference value will be observed in each layer.

Inverse D-Box Layer:

In the inverse D-Box layer of Step-5, three of four bundles are XORed with each other to construct the other bundle. Since Bundle-0 and Bundle-1 pairs are the same, after the inverse D-Box layer, Bundle-2 will have the difference of Bundle-3 and Bundle-3 will have the difference of Bundle-2. The pairs of Bundle-0 and Bundle-1 are still identical.

For Bundle-0: $P_{0,1}^{dbox} = P_{1,1} \oplus P_{2,1} \oplus P_{3,1}$ and $P_{0,2}^{dbox} = P_{1,2} \oplus P_{2,2} \oplus P_{3,2}$

$\Delta_0 = (P_{0,1}^{dbox} \oplus P_{0,2}^{dbox}) = (P_{1,1} \oplus P_{1,2}) \oplus (P_{2,1} \oplus P_{2,2}) \oplus (P_{3,1} \oplus P_{3,2}) = D$

For Bundle-1: $P_{1,1}^{dbox} = P_{0,1} \oplus P_{2,1} \oplus P_{3,1}$ and $P_{1,2}^{dbox} = P_{0,2} \oplus P_{2,2} \oplus P_{3,2}$

$\Delta_1 = (P_{1,1}^{dbox} \oplus P_{1,2}^{dbox}) = (P_{0,1} \oplus P_{0,2}) \oplus (P_{2,1} \oplus P_{2,2}) \oplus (P_{3,1} \oplus P_{3,2}) = D$

For Bundle-2: $P_{2,1}^{dbox} = P_{0,1} \oplus P_{1,1} \oplus P_{3,1}$ and $P_{2,2}^{dbox} = P_{0,2} \oplus P_{1,2} \oplus P_{3,2}$

$\Delta_2 = (P_{2,1}^{dbox} \oplus P_{2,2}^{dbox}) = (P_{0,1} \oplus P_{0,2}) \oplus (P_{1,1} \oplus P_{1,2}) \oplus (P_{3,1} \oplus P_{3,2}) = C$

For Bundle-3: $P_{3,1}^{dbox} = P_{0,1} \oplus P_{1,1} \oplus P_{2,1}$ and $P_{3,2}^{dbox} = P_{0,2} \oplus P_{1,2} \oplus P_{2,2}$

$\Delta_3 = (P_{3,1}^{dbox} \oplus P_{3,2}^{dbox}) = (P_{0,1} \oplus P_{0,2}) \oplus (P_{1,1} \oplus P_{1,2}) \oplus (P_{2,1} \oplus P_{2,2}) = B$

After the inverse S-Box layer, the difference of Bundle-0, Bundle-1 and Bundle-3 are still unknown. However, the output differences of the inverse S-Box layer are still the same for Bundle-0 and Bundle-1 because the same inputs produce the same outputs in the inverse S-Box layer. $P'_{0,1} = P'_{1,1}$ and $P'_{0,2} = P'_{1,2}$ is satisfied after the inverse S-Box layer. The difference value of Bundle-2 can be found by using *Undisturbed Bits* technique that is explained in Section 4.2. According to Table 4.4, the difference value of 27th column of Bundle-2 (1000) leads to (0??1) difference. Table 4.11 shows the difference values after the inverse D-Box and inverse S-Box operation

Table 4.11: Inverse D-Box and Inverse S-Box of Step-5

	STEP-5	
	Inverse D-Box	**Inverse S-Box**
Bundle-0	???????????????????????????????? ???????????????????????????????? ???????????????????????????????? ???????????????????????????????? Value is D	???????????????????????????????? ???????????????????????????????? ???????????????????????????????? ???????????????????????????????? Value is E
Bundle-1	???????????????????????????????? ???????????????????????????????? ???????????????????????????????? ???????????????????????????????? Value is D	???????????????????????????????? ???????????????????????????????? ???????????????????????????????? ???????????????????????????????? Value is E
Bundle-2	00001000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 Value is C	00000000000000000000000000000000 0000?0000000000000000000000000 0000?0000000000000000000000000 00001000000000000000000000000000 Value is H
Bundle-3	???????????????????????????????? ???????????????????????????????? ???????????????????????????????? ???????????????????????????????? Value is B	???????????????????????????????? ???????????????????????????????? ???????????????????????????????? ???????????????????????????????? Value is J

Round constant addition and L-Box Layer:

The round constant is added to bundles after the inverse S-Box layer. Recall that the round constant addition part does not change the difference value; however, for a bundle index *j*, the 4-bit round constant is XORed with the column index *j*. It means round constant is added to 3rd column of Bundle-3, 2nd column of Bundle-2, 1st column of Bundle-1 and 0th column of Bundle-0. Since round constant is added to different positions for each bundle, it is distributed to different positions for different bundles in the linear layer, which is inverse L-Box.

As can be seen in the Tables between 4.12 and 4.19, round constant bits are distributed to other bits in the inverse L-Box layer. Most of the bits are affected and they are marked as grey.

Table 4.12: Round constant addition for Bundle-0

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**

Table 4.13: The impact of round constant addition after inverse L-Box operation for Bundle-0

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**

Table 4.14: Round constant addition for Bundle-1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table 4.15: The impact of round constant addition after inverse L-Box operation for Bundle-1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table 4.16: Round constant addition for Bundle-2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table 4.17: The impact of round constant addition after inverse L-Box operation for Bundle-2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table 4.18: Round constant addition for Bundle-3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table 4.19: The impact of round constant addition after inverse L-Box operation for Bundle-3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

The impact of the distribution of round constant bits in the inverse L-Box layer shows that Bundle-0 and Bundle-1 pairs are not completely identical anymore after the inverse L-Box layer. However, round constant bits do not affect all 128 bits of a bundle. Some of the bits will not change and they are still identical for Bundle-0 and

Bundle-1 pairs. Note that **2nd, 10th, 21st, 22nd, 23rd** columns of the Bundle-0 and Bundle-1 pairs are not affected by round constant bits in the inverse L-Box part. In Table 4.20, the bold "?" bits of Bundle-0 and Bundle-1 differences are still unknown; however, it is proven that **2nd, 10th, 21st, 22nd, 23rd** columns of their pairs are identical. Also, the difference value of Bundle-0 and Bundle-1 is still the same and unknown. The similarity of bundle's pairs will be useful in the inverse D-Box part of Step-4 because it's expected that **2nd, 10th, 21st, 22nd, 23rd** columns of Bundle-0 and Bundle-1 pairs will still be the same after the inverse S-Box operation. Then, they will cancel out each other in the inverse D-Box part.

Table 4.20: Round constant addition and inverse L-Box of Step-5

	STEP-5	
	RC	**Inverse L-Box**
Bundle-0	???????????????????????????????? ???????????????????????????????? ???????????????????????????????? ???????????????????????????????? Value is E	????????**???**????????**?**????????**?**?? ????????**???**????????**?**????????**?**?? ????????**???**????????**?**????????**?**?? ????????**???**????????**?**????????**?**?? Value is K
Bundle-1	??????????????????????????????**?**? ??????????????????????????????**?**? ??????????????????????????????**?**? ??????????????????????????????**?**? Value is E	????????**???**????????**?**????????**?**?? ????????**???**????????**?**????????**?**?? ????????**???**????????**?**????????**?**?? ????????**???**????????**?**????????**?**?? Value is K
Bundle-2	00000000000000000000000000000000 0000?00000000000000000000000000 0000?00000000000000000000000000 00001000000000000000000000000000 Value is H	00000000**0**0**?**??00000??00**?**?00000??**?**0 ??00??0??**?**0**0**??00000???**0**000?000?**0**0 ??00??0??**?**0**1**??00001???**1**100?001?**1**0 11001101**1**0**0**??00000??1**0**??01000?**?**?
Bundle-3	???????????????????????????????? ???????????????????????????????? ???????????????????????????????? ???????????????????????????????? Value is J	???????????????????????????????? ???????????????????????????????? ???????????????????????????????? ???????????????????????????????? Value is M

According to Table 4.20, the difference value of Bundle-2 is distributed to other bits in the inverse L-Box layer. Some columns are still zero and some columns do not have any bits whose value is 1. It means these columns may be zero. On the other hand, columns with at least one "1" bit cannot have zero difference. This approach will be useful in the inverse S-Box layer.

Inverse S-Box and Inverse D-Box Layer:

After the inverse S-Box layer, the difference values of Bundle-0, Bundle-1 and Bundle-3 are still unknown. However, since **2nd, 10th, 21st, 22nd, 23rd** columns of

Bundle-0 and Bundle-1 pairs are identical, their difference values will still be identical after the inverse S-Box layer. The difference value of other columns may be different because inputs of the S-Boxes are different.

Table 4.21: Inverse S-Box of Step-5 and inverse D-Box of Step-4

	STEP-5	**STEP-4**
	Inverse S-Box	**Inverse D-Box**
Bundle-0	????????**???**??????????**?**????????**?**?? ????????**???**??????????**?**????????**?**?? ????????**???**??????????**?**????????**?**?? ????????**???**??????????**?**????????**?**?? Value is R_1	???????????????????????????????? ???????????????????????????????? ???????????????????????????????? ???????????????????????????????? Value is Y
Bundle-1	????????**???**??????????**?**????????**?**?? ????????**???**??????????**?**????????**?**?? ????????**???**??????????**?**????????**?**?? ????????**???**??????????**?**????????**?**?? Value is R_2	???????????????????????????????? ???????????????????????????????? ???????????????????????????????? ???????????????????????????????? Value is Z
Bundle-2	**xx**00**xx**0**xx**0**x**??0000**x**??**xxx**?0**x**00**x**?**x**? **xx**00**xx**0**xx**0**x**??0000**x**??**xxx**?0**x**00**x**?**x**? **xx**00**xx**0**xx**0**x**??0000**x**??**xxx**?0**x**00**x**?**x**? **xx**00**xx**0**xx**0**x**??0000**x**??**xxx**?0**x**00**x**?**x**? Value is S	???????????????????????????????? ???????????????????????????????? ???????????????????????????????? ???????????????????????????????? Value is T_1
Bundle-3	???????????????????????????????? ???????????????????????????????? ???????????????????????????????? ???????????????????????????????? Value is T	??????**x**?**x**??????????**x**???????????? ??????**x**?**x**??????????**x**???????????? ??????**x**?**x**??????????**x**???????????? ??????**x**?**x**??????????**x**???????????? Value is S_1 (Nonzero)

For Bundle-2, some bits are represented "x" because the input difference of inverse S-Box contains at least one "1" bit. It is obvious that the output difference of inverse S-Box will contain at least one "1" bit. It means one of the "x" values must be "1" for each column that includes "x."

If $Inv - SBox(?\,010) = (xxxx)$, at least one of "$x$" is 1.

As mentioned in Section 2.4.4, in the D-Box layer, three of four bundles are XORed with each other to construct the other bundle. To construct Bundle-3; Bundle-0, Bundle-1 and Bundle-2 are XORed with each other. It is known that **2nd, 10th, 21st, 22nd, 23rd** columns of Bundle-0 and Bundle-1 pairs are exactly the same. When Bundle-0 and Bundle-1 are XORed with each other, **2nd, 10th, 21st, 22nd, 23rd** columns will be zero. Therefore, the difference of Bundle-3 will come from Bundle-2 on these columns after the inverse D-Box layer. As can be seen in Table 4.21, **23rd, 22nd, 10th** columns of Bundle-2 contain at least one "1" bit. It means the difference of

Bundle-3, which is "S_1" cannot be zero after the inverse D-Box layer with probability one.

By this point, firstly, the 5-Step truncated differential that starts from Step-2 was explained and then the output difference of Super S-Box of Step-4, which is $(\phi, \psi, \omega, 0)$ was obtained with probability one. Then, one step and an inverse D-Box operation in the backward direction were added to the bottom of this differential and the output difference of inverse D-Box of Step-4, which is (Y, Z, T_1, S_1) was obtained with probability one. The idea is to find a nonzero difference value of Bundle-3, which is represented as "S_1" so that output difference of Super S-Box and output difference of inverse D-Box cannot match in the middle of Step-4. As it is explained in detail, in the Step-4, the difference of Bundle-3, which is "S_1" cannot be zero if the output difference of Step-5 is (A, A, B, C) . It can be said that the output difference of Step-5, which is (A, A, B, C) cannot come from the input difference of Step-0 which is $(\lambda, \mu, \nu, 0)$ if the output difference of Step-2 Super S-Box operation is $(\beta, \beta, \beta, 0)$. Thus, 6-Step impossible differential distinguisher is obtained.

$$\Delta_o = (A, A, B, C) \nleftarrow \Delta_i = (\lambda, \mu, \nu, 0)\ with\ probability\ 1$$

To sum up, we have defined the output difference of Step-5 as (A, A, B, C). Since the difference value of "A" and "B" can be any value and 128-bit of "C" is fixed, there are 2^{256} possible output differences that can be generated. Therefore, this impossible differential holds with the probability 2^{-256} for a random permutation. Due to the birthday paradox, the impossible differential distinguisher needs 2^{128} different pairs to distinguish Shadow-512 from a random permutation. According to (Derbez et al., 2020), the probability of their 6-Step truncated differential distinguisher is $2^{-16.245}$. Compared to their result, our distinguisher works with probability one. Although our distinguisher has a better probability, a distinguishing attack on a random permutation would require around 2^{128} pairs due to the filtering conditions on the input pairs.

4.4. 7- Step Impossible Differential Distinguisher of Shadow-512

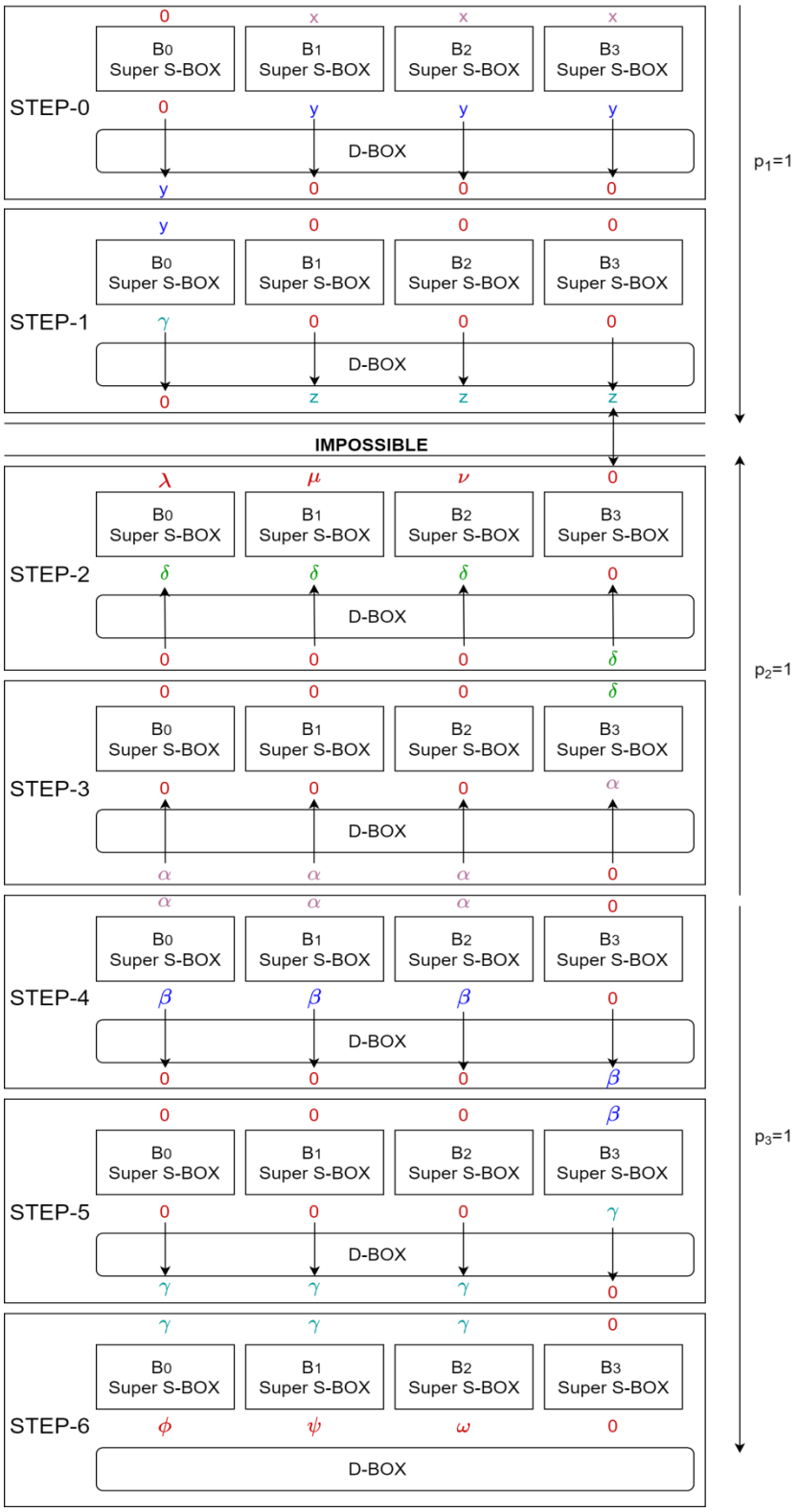

Figure 4.3: 7-Step Impossible Differential Distinguisher of Shadow-512

Normally, Shadow-512 is designed as a 6-Step permutation. However, it can be thought of as a 7-step design as if it is a round-extended version. In this section, 7-step impossible differential distinguisher will be explained. The 7-step impossible differential distinguisher is found by adding two more steps in the forward direction on the top of 5-Step truncated differential that was explained in Section 3.2. The 5-step truncated differential starts from Step-4 and consists of two steps in the forward direction and two steps in the backward direction. After that, another two steps are added to the top of the 5-Step truncated differential and it starts from Step-0. If the difference value at the end of Step-1 and the difference value at the beginning of Step-2 do not match, the 7-Step impossible differential distinguisher will be obtained. The 7-Step impossible differential distinguisher is shown in Fig. 4.3.

Let σ_j denote the Super S-Box operation and j denotes the bundle index.

Algorithm − 5 7 − Step impossible distinguisher

Impossible: The difference of Step-2 which is $(\lambda, \mu, \nu, 0)$ does not match the difference of Step-1 $(0, z, z, z)$ in the middle.

i. Choose a random pair such that $m \in \mathbb{F}_2^{128}$ and $(m \oplus \beta) \in \mathbb{F}_2^{128}$ in Step-4. β must be zero on 0th, 1st, 2nd, 3rd columns of a bundle.

ii. Choose a random state $b \in \mathbb{F}_2^{128}$.

iii. Compute $n = \sigma_j^{-1}(m)$ and $n + \alpha = \sigma_j^{-1}(m + \beta)$ for $0 \leq j \leq 2$. Set the states at Step-4 such that

$$X_4 = (n, n, n, b) \; and \; X_4' = (n + \alpha, n + \alpha, n + \alpha, b) \rightarrow \Delta = (\alpha, \alpha, \alpha, 0)$$

iv. Iterate Step-5 and Step-6 on X_4 and X_4' to obtain $(\phi, \psi, \omega, 0)$ in Step-6.

v. The difference of Step-6 $(\phi, \psi, \omega, 0)$ cannot come from the input difference $(0, x, x, x)$ in Step-0 if the difference of Step-3 is $(\beta, \beta, \beta, 0)$.

According to Algorithm-5, in the beginning, 5-Step truncated differential starts from Step-4 with the difference value $(\alpha, \alpha, \alpha, 0)$. The pairs of Bundle-0, Bundle-1 and Bundle-2 are in a 3-Identical state. The 3-Identical state cannot be preserved after the first round constant addition since round constant is added to different columns of different bundles. However, after one step, three identical differences can be obtained by choosing the right difference value of α. As mentioned in Section 3.2, to obtain $\alpha \rightarrow \beta$ transition after the Super S-Box, the difference β must be set to zero for columns that round constant is added. In other words, the difference value of 0th, 1st,2nd,3rd columns of a bundle must be set to zero for β. Therefore after the Super S-Box, the difference will become $(\beta, \beta, \beta, 0)$ and three identical differences are obtained. After D-Box, the difference will become $(0,0,0,\beta)$ since three of four bundles are XORed with each other in the D-Box layer. The difference $(0,0,0,\beta)$ will become $(0,0,0,\gamma)$ after Super S-Box of Step-5 because zero input difference leads to zero output difference in the Super S-Box layer. In the D-Box layer of Step-5, the difference $(0,0,0,\gamma)$ will become $(\gamma, \gamma, \gamma, 0)$ since "γ" difference of Bundle-3 is distributed to other bundles. In the Super S-Box layer of Step-6, input differences

of Bundle-0, Bundle-1 and Bundle-2, which are "γ" will lead to different values for different bundles. Although Bundle-0, Bundle-1 and Bundle-2 have the same difference value before the Super S-Box layer, their pairs are not the same. The same input differences can lead to different output differences in the Super S-Box layer. Therefore the difference $(\gamma, \gamma, \gamma, 0)$ will become $(\phi, \psi, \omega, 0)$ with probability one after the Super S-Box layer of Step-6. The difference values of the first three bundles can be any value, but the difference of Bundle-3 is certainly zero. Now, we are going back to Step-4, where iteration starts. Two steps of the inverse operation are applied to $(\alpha, \alpha, \alpha, 0)$ the difference in Step-4. After the inverse D-Box layer of Step-3, the difference will become $(0,0,0, \alpha)$. In inverse Super S-Box of Step-3, the difference $(0,0,0, \alpha)$ will become $(0,0,0, \delta)$ and the difference $(0,0,0, \delta)$ will become $(\delta, \delta, \delta, 0)$ after inverse D-Box layer. Although the differences of Bundle-0, Bundle-1 and Bundle-2 are the same, their pairs may not be identical. Since the same input difference can lead to different output difference in the inverse Super S-Box layer, the difference $(\delta, \delta, \delta, 0)$ will become $(\lambda, \mu, \nu, 0)$ after inverse Super S-Box of Step-2. The difference value of Bundle-0, Bundle-1 and Bundle-2 can be any value, but the difference of Bundle-3 is definitely zero after two steps in the backward direction if the input difference is $(\alpha, \alpha, \alpha, 0)$ in Step-4.

As mentioned above, the difference value at the beginning of Step-2 is$(\lambda, \mu, \nu, 0)$ and it is proven that the difference value of Bundle-3 is zero. The idea is adding two more steps to the top of the difference value $(\lambda, \mu, \nu, 0)$ in Step-2. After two steps that are added to the top, if it is proven that the difference value of Bundle-3 of Step-1 cannot be zero, it cannot match the difference value at the beginning of Step-2. Thus, 7-step impossible differential distinguisher will be obtained.

To find impossible differential distinguisher, the difference value $(0, x, x, x)$ is given to bundles in Step-0. The pairs are built in the same way as in Step-4 but now zero difference is given to Bundle-0 instead of Bundle-3. It means Bundle-1, Bundle-2 and Bundle-3 should be in the 3-Identical state. The 3-Identical state cannot be preserved after the round constant addition part in the Super S-Box layer. However, if the output difference of Super S-Box, which is "y", is set to zero on the columns that round constants are added, the $x \rightarrow y$ level transition is valid for Bundle-1, Bundle-2 and Bundle-3. Thus the round constant addition that depends on bundle index cannot affect the output differences of Super S-Box. The input difference $(0, x, x, x)$ will become $(0, y, y, y)$ after Super S-Box of Step-0. After that, the input difference value of the D-Box $(0, y, y, y)$ will become $(y, 0,0,0)$ since three of four bundles are XORed with each other to construct the other bundle. In Step-1, the input difference $(y, 0,0,0)$ will become $(z, 0,0,0)$ after Super S-Box. Bundle-0 difference "z"will distribute to the other three bundles and the difference value will become $(0, z, z, z)$ in the D-Box layer of Step-1. It is certain that the difference value of Bundle-3 "z" is nonzero.

Therefore, the nonzero difference of Bundle-3, which is "z" cannot match the zero difference of Bundle-3 at the beginning of Step-2. It can be said that the output difference of Step-6 $(\phi, \psi, \omega, 0)$ cannot come from the input difference of Step-0 $(0, x, x, x)$ with probability one after 7-Step iteration if the difference of Step-4 is $(\beta, \beta, \beta, 0)$.

Since the idea is to make the output difference value of Bundle-3 nonzero after Step-1, there are three input difference structures that can be given to the bundles at the beginning of Step-0. In Table 4.22, all possible input difference structures that lead to nonzero Bundle-3 difference after Step-1 are shown.

Table 4.22: The possible input difference structures of 7-Step impossible differential distinguisher

INITIALIZATION	$(0, x, x, x)$	$(x, 0, x, x)$	$(x, x, 0, x)$
STEP-0	$(y, 0, 0, 0)$	$(0, y, 0, 0)$	$(0, 0, y, 0)$
STEP-1	$(0, z, z, z)$	$(z, 0, z, z)$	$(z, z, 0, z)$

7-Step impossible differential characteristics:

$\Delta_o = (\phi, \psi, \omega, 0) \nleftarrow \Delta_i = (0, x, x, x)$ *with probability* 1 or

$\Delta_o = (\phi, \psi, \omega, 0) \nleftarrow \Delta_i = (x, 0, x, x)$ *with probability* 1 or

$\Delta_o = (\phi, \psi, \omega, 0) \nleftarrow \Delta_i = (x, x, 0, x)$ *with probability* 1

In Step-0, we use two 3-Identical input messages $P_1 = (N, M_1, M_1, M_1)$ and $P_2 = (N, M_2, M_2, M_2)$ such that $P_1 \oplus P_2 = (0, x, x, x)$. The probability of constructing P_1 is 2^{-256} because 256 bits are fixed. The probability of constructing P_2 is $2^{-256} \times 2^{-128}$ because the Bundle-3 of P_2 should be the same as the Bundle-3 of P_1. Therefore, we can say that the output difference $(\phi, \psi, \omega, 0)$ of Step-6 cannot come from the input difference $(0, x, x, x)$ of Step-0 with probability 2^{-640} for a random permutation. Due to the birthday paradox, one needs at least 2^{320} different input pairs to observe $(\phi, \psi, \omega, 0)$ output difference at the end of Step-6.

On the other hand, we can also build this 7-Step impossible differential by not using the 3-Identical property. We need to obtain $(0, y, y, y)$ difference right after the Super S-Box of Step-0. If we start from the D-Box operation of Step-0 with the input difference $(0, y, y, y)$, we can say that the output difference value of Step-6 $(\phi, \psi, \omega, 0)$ cannot come from the input difference of Step-0 which is $(0, y, y, y)$. This 7-Step impossible differential distinguisher holds with the probability $p = 2^{-384}$ for a random permutation since the 128 bits of input differences are fixed. Therefore the data complexity of the 7-Step impossible differential distinguisher is 2^{192} pairs because of the birthday paradox.

To sum up, if Shadow-512 permutation were designed as 7-Step, there exists an impossible differential that distinguishes 7-Step Shadow-512 from random permutation. According to (Derbez et al., 2020), the probability of their 6-Step truncated differential distinguisher is $2^{-16.245}$. Compared to their result, our distinguisher works with probability one. Although our distinguisher has a better probability, a distinguishing attack on a random permutation would require around 2^{192} pairs.

4.5. 8-Step Improbable Differential Distinguisher of Shadow-512

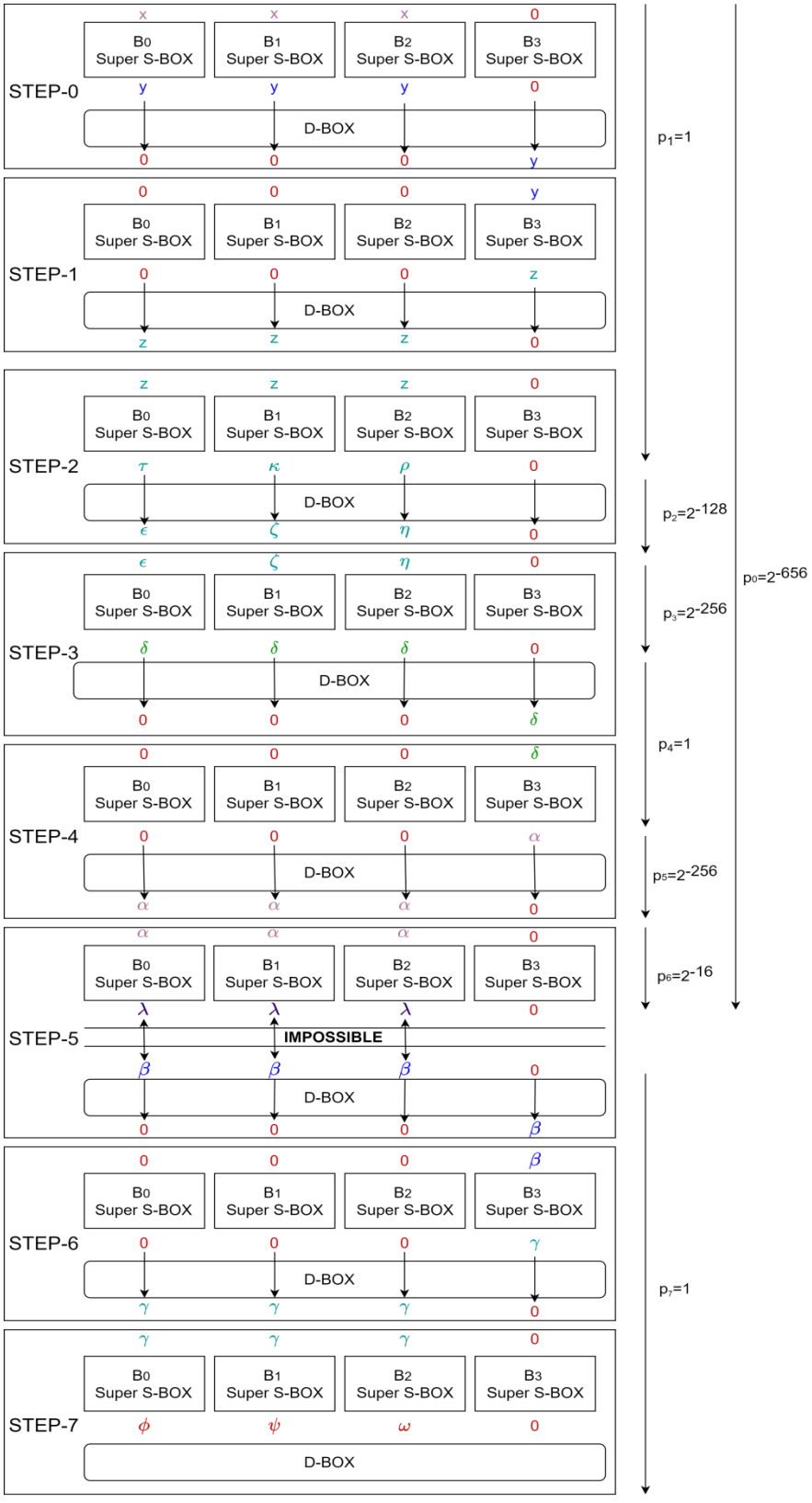

Figure 4.4: 8-Step Improbable Differential Distinguisher of Shadow-512

In Section 4.3, two different 6-Step impossible differential distinguishers of Shadow-512 were explained. As mentioned in Section 2.3, Shadow-512 was designed as 6-Step; however, it is possible to find a distinguisher that covers more than six steps. In Section 4.4, we have introduced a 7-Step impossible differential as if Shadow-512 has been designed as seven steps. Similarly, it can be considered that Shadow-512 consists of eight steps. In this section, 8-Step improbable differential distinguisher will be explained. The 8-Step improbable differential has two parts. Firstly, three steps that start from Step-5 will be described. Then, five steps that start from Step-0 will be explained. The improbable differential distinguisher will be obtained between Step-4 and Step-5.

Let σ_j denote the Super SBOX operation and j denotes the bundle index.

Algorithm − 6 8 − Step improbable distinguisher

Improbable: The difference of Step-5 $(\phi, \psi, \omega, 0)$ cannot come from the input difference $(x, x, x, 0)$ in Step-0 with the probability 1-2^{-656} if the difference of Step-5 is $(\beta, \beta, \beta, 0)$.

i. In Step-5, choose a pair such that $m \in \mathbb{F}_2^{128}$ and $(m \oplus \beta) \in \mathbb{F}_2^{128}$ before $\mathrm{D-BOX}$. β must be zero on the 0th, 1st, 2nd, 3rd columns of a bundle.

ii. Choose a random state $b \in \mathbb{F}_2^{128}$.

iii. Compute $n = \sigma_j^{-1}(f)$ and $n + \alpha = \sigma_j^{-1}(m + \beta)$ for $0 \leq j \leq 2$. Set the states at Step-5 such that

$$X_5 = (n, n, n, b) \; and \; X_5' = (n + \alpha, n + \alpha, n + \alpha, b) \rightarrow \Delta = (\alpha, \alpha, \alpha, 0)$$

iv. Iterate Step-6 and Step-7 to obtain $(\phi, \psi, \omega, 0)$ output difference.

v. If the input difference is $(x, x, x, 0)$ in Step-0, the output difference can be $(\phi, \psi, \omega, 0)$ in Step-7 with the very low probability 2^{-656}.

According to Algorithm-6, the improbable differential starts from the D-Box operation of Step-5 with the difference value $(\beta, \beta, \beta, 0)$. The difference "β" must be set to zero on the 0th,1th,2nd,3rd columns that round constant is added. To obtain 3-Identical State at the input of Super S-Box of Step-5, "β" must be set to zero on these bits after the Super S-Box operation. In the D-Box layer of Step-5, three of four bundles are XORed with each other and the difference value will become $(0,0,0, \beta)$. In Step-6, $(0,0,0, \beta)$ input difference will become $(0,0,0, \gamma)$ after Super S-Box layer. The difference "γ" will be distributed to other bundles in the D-Box layer of Step-6. Step-7 starts with the difference value $(\gamma, \gamma, \gamma, 0)$ and $(\gamma, \gamma, \gamma, 0)$ the difference will become $(\phi, \psi, \omega, 0)$ after Super S-Box of Step-7 with probability one.

The second part starts from Step-0 with the difference value $(x, x, x, 0)$. Bundle-0, Bundle-1 and Bundle-2 pairs must be 3-Identical. The 3-Identical state cannot be preserved after the first round constant addition part. Therefore, to obtain three identical output differences of Super S-Box, the output difference of Super S-Box, which is "y" must be set to zero on columns that round constants are added. The

detailed explanation about building pairs that gives $x \longrightarrow y$ transition was explained in Section 3.2. Thus, the input difference of Step-0, which is $(x, x, x, 0)$ will become $(y, y, y, 0)$ after the Super S-Box of Step-0. After the D-Box of Step-0, since the differences of Bundle-0, Bundle-1 and Bundle-2 are the same, their differences cancel out each other and the output difference of the D-Box will become $(0,0,0, y)$. In Step-1, the difference $(0,0,0, y)$ will become $(0,0,0, z)$ after the Super S-Box layer and the difference $(0,0,0, z)$ will become $(z, z, z, 0)$ after the D-Box because the difference of Bundle-3, which is "z" will be distributed to other bundles. After the Super S-Box of Step-2, the difference $(z, z, z, 0)$ will become $(\tau, \kappa, \rho, 0)$. It means the differences of Bundle-0, Bundle-1 and Bundle-2 can be any value. Since round constant is added to different columns for different bundles, the output difference of S-Box can be different for the same input. In other words, although the round constant addition part does not affect the difference value, it affects the values of input pairs. Therefore, even if output differences of Bundle-0, Bundle-1 and Bundle-2 are the same after the RC part, their input pairs are different from each other. After all, the second S-Box layer that is done right after the round constant addition part may give different output differences for the same input differences according to DDT. Up to now, it is guaranteed that if the input difference is $(x, x, x, 0)$ in the beginning of Step-0, the output difference of the Super S-Box of Step-2 will be $(\tau, \kappa, \rho, 0)$ with probability $p_1 = 1$.

In the D-Box layer of Step-2, Bundle-0, Bundle-1 and Bundle-2 are XORed with each other to construct the difference of Bundle-3. As explained above, the difference values of Bundle-0, Bundle-1 and Bundle-2 are "τ" , "κ" and " ρ" , respectively. Since the exact values of "τ", "κ" and" ρ" are not known, they might be anything except for zero. Therefore, if they are XORed with each other in the D-Box, the result can be zero with the probability p_2.

After the D-Box layer of Step-2:

$\Delta_0^{DBOX} = \kappa \oplus \rho \oplus 0 = \epsilon$

$\Delta_1^{DBOX} = \tau \oplus \rho \oplus 0 = \zeta$

$\Delta_2^{DBOX} = \tau \oplus \kappa \oplus 0 = \eta$

$\Delta_3^{DBOX} = \tau \oplus \kappa \oplus \rho = \mathbf{0} \quad with\ probability\ p_2$

Since 128 bits are fixed to zero at the difference of Bundle-3, the probability of having zero difference of Bundle-3 is 2^{-128}. $p_2 = 2^{-128}$.

At the beginning of Step-3, the differences are $(\epsilon, \zeta, \eta, 0)$. It is obvious that zero input difference leads to zero output difference after the Super S-Box of Step-3 for Bundle-3. The purpose is to obtain the same output differences for Bundle-0, Bundle-1 and Bundle-2.

After the Super S-Box layer of Step-3:

$\Delta_0^{input} = \epsilon \longrightarrow \Delta_0^{Super\ SBOX} = \delta$

$\Delta_1^{input} = \zeta \longrightarrow \Delta_1^{Super\ SBOX} = \delta$

$\Delta_2^{input} = \eta \longrightarrow \Delta_2^{Super\ SBOX} = \delta$

$\Delta_3^{input} = 0 \longrightarrow \Delta_3^{Super\ SBOX} = 0$

The probability of having identical differences of Bundle-0, Bundle-1 and Bundle-2 is 2^{-256} because the difference of Bundle-3 will be definitely zero and there are 2^{128} different "δ" values of whole 2^{384} structures. $p_3 = 2^{-256}$.

After the D-Box of Step-3, the difference $(\delta, \delta, \delta, 0)$ will become $(0,0,0, \delta)$ since three of four bundles will be XORed with each other. In Step-4, after the Super S-Box layer, the difference will become $(0,0,0, \alpha)$ with probability one. $p_4 = 1$.

After the D-Box Layer of Step-4

After the D-Box layer of Step-4, the difference $(0,0,0, \alpha)$ will be definitely $(\alpha, \alpha, \alpha, 0)$. Although the differences of Bundle-0, Bundle-1 and Bundle-2 are the same value, their input pairs may be different from each other. The idea is obtaining 3-Identical State at the beginning of Step-5. In other words, pairs of Bundle-0, Bundle-1 and Bundle-2 should be the same. The probability of having a 3-Identical state is 2^{-256} because 128 bits are fixed for a bundle and the same value will be given to the other two bundles. $p_5 = 2^{-256}$

After the Super S-Box Layer of Step-5

At the beginning of Step-5, Bundle-0, Bundle-1 and Bundle-2 pairs are the same and their differences are $(\alpha, \alpha, \alpha, 0)$. It is known that the only difference between the Super S-Boxes is round constant operation. To obtain $\alpha \longrightarrow \lambda$ transition for each Bundle-0, Bundle-1 and Bundle-2 differences, λ should be zero on columns that round constant is added. The other columns will definitely be the same for the first three bundles after the Super S-Box operation. The probability of having zero difference on 0^{th},1^{th},2^{nd},3^{rd} columns of "λ" is $p_6 = 2^{-16}$ since 16 bits are fixed to zero. Thus, it would be said that the difference "λ" is equal to "β" after the Super S-Box of Step-5 with probability $p_0 = p_1 \cdot p_2 \cdot p_3 \cdot p_4 \cdot p_5 \cdot p_6$. Therefore, the probability of not having the output difference $(\beta, \beta, \beta, 0)$ after Super S-Box of Step-5 is $1 - 2^{-656}$ if the input difference is $(x, x, x, 0)$. It is obvious that the differences $(\lambda, \lambda, \lambda, 0)$ and $(\beta, \beta, \beta, 0)$ miss in the middle of Step-5 with the probability $1 - 2^{-656}$.

$\Delta^{Super\ S-BOX} = (\lambda, \lambda, \lambda, 0) \nrightarrow (\beta, \beta, \beta, 0)\ with\ probability\ p' = 1 - 2^{-656}$.

The probability p' shows that it is not impossible, but it is improbable. The probability of not having $(\beta, \beta, \beta, 0)$ the difference after five and a half steps is

almost one. Therefore it can be said that the output difference of Step-7 which is $(\phi, \psi, \omega, 0)$ cannot come from the input difference of Step-0 which is$(x, x, x, 0)$ with the probability $p' = 1 - 2^{-656}$.

$$\Delta_o = (\phi, \psi, \omega, 0) \nleftarrow \Delta_i = (x, x, x, 0)\ with\ probability\ 1 - 2^{-656}$$

In Step-0, we use two 3-Identical input messages $P_1 = (M_1, M_1, M_1, N)$ and $P_2 = (M_2, M_2, M_2, N)$ such that $P_1 \oplus P_2 = (x, x, x, 0)$. The probability of constructing P_1 is 2^{-256} because 256 bits are fixed. The probability of constructing P_2 is $2^{-256} \times 2^{-128}$ because the Bundle-3 of P_2 should be the same as the Bundle-3 of P_1. Therefore, we can say that the output difference $(\phi, \psi, \omega, 0)$ of Step-7 can come from the input difference $(x, x, x, 0)$ with the probability $p = 2^{-640}$ for a random permutation. Due to the birthday paradox, one needs at least 2^{320} different input pairs to observe $(\phi, \psi, \omega, 0)$ output difference at the end of Step-7.

On the other hand, we can also build this 8-Step improbable differential distinguisher by starting after the Super S-Box layer of Step-0 with the difference $(y, y, y, 0)$. Now, we do not use the 3-Identical property and Bundle-0, Bundle-1 and Bundle-2 may be different from each other at the beginning of Step-0. However, their differences should be the same right after the Super S-Box of Step-0. Therefore, we can still say that the output difference of Step-7 which is $(\phi, \psi, \omega, 0)$ cannot come from the input difference of Step-0 which is $(y, y, y, 0)$ with the probability $p' = 1 - 2^{-656}$.

$$\Delta_o = (\phi, \psi, \omega, 0) \nleftarrow \Delta_i = (y, y, y, 0)\ with\ probability\ 1 - 2^{-656}$$

The probability of observing $(y, y, y, 0)$ input difference at Step-0 is $p = 2^{-384}$ for a random permutation because 128 bits are fixed at the input. Due to the birthday paradox, one needs at least 2^{192} different input pairs to observe $(\phi, \psi, \omega, 0)$ output difference at the end of Step-7.

In a nutshell, It can be seen that the probability of improbable differential distinguisher is less than the probability of random permutation in both scenarios. As mentioned in Section 1.6, since $p_0 < p$, this improbable differential distinguisher is valid for 8-Step Shadow-512. Our 8-Step improbable differential distinguisher is the longest differential distinguisher of Shadow-512 that has been found yet.

CHAPTER 5

CONCLUSION

Differential cryptanalysis is one of the most common methods that help an attacker exploit the differential relations between inputs and outputs of an algorithm for a specific round. It investigates the output differences when specific input differences are given to inputs. An output of a cryptographic algorithm should seem random. However, it is possible to distinguish the algorithm's output from a random permutation by using differential cryptanalysis techniques.

In this thesis, we have worked on the Spook algorithm, which is one of the round 2 candidates of the NIST's lightweight cryptography competition. Spook algorithm uses Shadow-512 as an internal permutation. Since NIST encouraged the public evaluation of candidate algorithms, we have tried finding differential distinguishers of Shadow-512 that help us distinguish Shadow-512 from a random permutation. The authors of Spook recommended using Shadow-512 as 6-Step. They also stated that it could be used as 4-Step. For this purpose, we have worked on the concept of impossible and improbable differential cryptanalysis. We have investigated the *undisturbed bits* of Shadow's S-Box to find longer impossible differentials. In (Derbez et al., 2020), they have already found a 5-Step truncated differential distinguisher of Shadow-512 with probability one. We have tried finding impossible differential distinguishers of Shadow-512 that cover more steps than their differentials. In a nutshell, we have found two different 6-Step impossible differential distinguishers that cover full Shadow permutation by using their 5-Step truncated differential. In addition, in (Derbez et al., 2020), they found 7-Step truncated differential distinguisher with probability $2^{-16.245}$ as if Shadow-512 has a 7-Step. From this point of view, we have tried finding longer differentials of Shadow-512. We have found 7-Step impossible differential distinguisher with probability one. Besides, we have found an 8-Step improbable differential distinguisher of Shadow-512 with probability 2^{-656}. We have found the longest differential distinguisher of Shadow-512.

Table 5.1: Summary of truncated, impossible, and improbable differential distinguishers on Shadow-512

Steps	**Method**	**Probability for Shadow-512 permutation**	**Probability for random permutation**	**Section**
5	Truncated Diff.	1	2^{-128}	(Derbez et al., 2020)
6	Truncated Diff.	$2^{-16.245}$	2^{-128}	(Derbez et al., 2020)
6	Impossible Diff.	0	2^{-128}	4.3.1
6	Impossible Diff.	0	2^{-256}	4.3.2
7	Truncated Diff.	$2^{-16.245}$	2^{-128}	(Derbez et al., 2020)
7	Impossible Diff.	0	2^{-640} or 2^{-384}	4.4
8	Improbable Diff.	2^{-656}	2^{-640} or 2^{-384}	4.5

To sum up, it would have been expected that the output of the Shadow-512 should seem random for its security. However, we have proven that the Shadow-512 has non-random behavior even if it is extended to eight steps. We are able to distinguish 6-,7-,8-Step Shadow-512 from a random permutation. We cannot perform a forgery attack on Spook by using our distinguishers. To perform a forgery attack on Spook, we need to find a collision at the output to produce the same tag. Since we found the impossible and improbable distinguishers, we cannot specify the exact output differences that should be obtained at the output. We can only specify which input differences do not lead to which output differences.

REFERENCES

Albrecht, M. R., Driessen, B., Kavun, E. B., Leander, G., Paar, C., & Yalçin, T. (2014). Block ciphers - Focus on the linear layer (feat. PRIDE). In J. A. Garay & R. Gennaro (Eds.), *Advances in Cryptology – CRYPTO 2014. CRYPTO 2014. Lecture Notes in Computer Science, vol 8616.* (pp. 57–76). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-44371-2_4

Araki, K., Satoh, T., & Miura, S. (1998). Overview of elliptic curve cryptography. In H. Imai & Y. Zheng (Eds.), *Public Key Cryptography. PKC 1998.Lecture Notes in Computer Science, vol 1431*. (pp. 29–49). Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0054012

Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., & Regazzoni, F. (2015). Midori: A Block Cipher for Low Energy. In T. Iwata & J. H. Cheon (Eds.), *Advances in Cryptology – ASIACRYPT 2015. Lecture Notes in Computer Science, vol 9453.* (pp. 411–436). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-48800-3_17

Banik, S., Chakraborti, A., Iwata, T., Minematsu, K., Nandi, M., Peyrin, T., Sasaki, Y., Sim, S. M., & Todo, Y. (2019). GIFT-COFB. *In: Lightweight Cryptography Standardization Process Round 2 Submission, NIST*. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/gift-cofb-spec-round2.pdf

Bao, Z., Chakraborti, A., Datta, N., Guo, J., Nandi, M., Peyrin, T., & Yasuda, K. (2019). *PHOTON-Beetle Authenticated Encryption and Hash Family*. In: Lightweight Cryptography Standardization Process Round 2 Submission, NIST. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/photon-beetle-spec-round2.pdf

Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., & Wingers, L. (2015). Simon and Speck: Block Ciphers for the Internet of Things. *Proceedings of the 52nd Annual Design Automation Conference on - DAC '15, July*, 1–6. http://dl.acm.org/citation.cfm?doid=2744769.2747946

Beierle, C., Biryukov, A., Santos, L. C. dos, Großschadl, J., Perrin, L., Udovenko, A., Velichkov, V., & Wang, Q. (2019). Schwaemm and Esch : Lightweight Authenticated Encryption and Hashing using the Sparkle Permutation Family. *In: Lightweight Cryptography Standardization Process Round 2 Submission, NIST*. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/sparkle-spec-round2.pdf

Beierle, C., Canteaut, A., Leander, G., & Rotella, Y. (2017). Proving resistance against invariant attacks: How to choose the round constants. In J. Katz & H. Shacham (Eds.), *Advances in Cryptology – CRYPTO 2017. CRYPTO 2017. Lecture Notes in Computer Science, vol 10402.* (pp. 647–678). Springer, Cham. https://doi.org/10.1007/978-3-319-63715-0_22

Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., & Sim, S. M. (2016). The SKINNY family of block ciphers and its low-latency variant MANTIS. In M. Robshaw & J. Katz (Eds.), *Advances in Cryptology – CRYPTO 2016. CRYPTO 2016. Lecture Notes in Computer Science, vol 9815.* (pp. 123–153). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-53008-5_5

Bellizia, D., Berti, F., Bronchain, O., Cassiers, G., Duval, S., Guo, C., Leander, G., Leurent, G., Levi, I., Momin, C., Pereira, O., Peters, T., Standaert, F.-X., Udvarhelyi, B., & Wiemer, F. (2020). *Spook: Sponge-Based Leakage-Resistant Authenticated Encryption with a Masked Tweakable Block Cipher*. https://www.spook.dev/assets/TOSC_Spook.pdf

Bellizia, D., Berti, F., Bronchain, O., Cassiers, G., Duval, S., Guo, C., Leander, G., Leurent, G., Levi, I., Momin, C., Pereira, O., Peters, T., Standaert, F. X., Udvarhelyi, B., & Wiemer, F. (2019). Spook: Sponge-based leakage-resistant authenticated encryption with a masked tweakable block cipher. *In: Lightweight Cryptography Standardization Process Round 2 Submission, NIST*. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/Spook-spec-round2.pdf

Berti, F., Guo, C., Pereira, O., Peters, T., & Standaert, F.-X. (2019). TEDT, a Leakage-Resilient AEAD Mode for High Physical Security Applications. *IACR Cryptology EPrint Archive, Report 2019/137*. https://doi.org/10.46586/tches.v2020.i1.256-320

Berti, F., Pereira, O., Peters, T., & Standaert, F.-X. (2017). On Leakage-Resilient Authenticated Encryption with Decryption Leakages. *IACR Transactions on Symmetric Cryptology*, *2017(3)*(3), 271–293. https://doi.org/10.46586/tosc.v2017.i3.271-293

Bertoni, G., Daemen, J., & Peeters, M. (2009). Keccak sponge function family main document. *Submission to NIST, Round 2*.

Bertoni, G., Daemen, J., Peeters, M., & Van Assche, G. (2007). Sponge functions. In *Ecrypt Hash Workshop 2007*.

Bertoni, G., Daemen, J., Peeters, M., & Van Assche, G. (2012). Duplexing the sponge: Single-pass authenticated encryption and other applications. In A. Miri & S. Vaudenay (Eds.), *Selected Areas in Cryptography. SAC 2011. Lecture Notes in Computer Science, vol 7118* (pp. 320–337). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-28496-0_19

Beyne, T., Mennink, B., Chen, Y. L., & Dobraunig, C. (2019). Elephant v1.1. *In: Lightweight Cryptography Standardization Process Round 2 Submission, NIST*, 1–48. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/

documents/round-2/spec-doc-rnd2/elephant-spec-round2.pdf

Biham, E., Anderson, R., & Knudsen, L. (1998). Serpent: A New Block Cipher Proposal. In S. Vaudenay (Ed.), *Fast Software Encryption. FSE 1998. Lecture Notes in Computer Science, vol 1372* (pp. 222–238). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-69710-1_15

Biham, E., Biryukov, A., & Shamir, A. (1998). Impossible Differential Attacks. In *Rump Session of CRYPTO 1998*.

Biham, E., Biryukov, A., & Shamir, A. (1999). Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In J. Stern (Ed.), *Advances in Cryptology — EUROCRYPT '99. EUROCRYPT 1999. Lecture Notes in Computer Science, vol 1592* (pp. 12–23). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-48910-X_2

Biham, E., & Shamir, A. (1991). Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, *4*(1), 3–72. https://doi.org/10.1007/BF00630563

Biryukov, Alex, & Perrin, L. (2017). State of the Art in Lightweight Symmetric Cryptography. *Cryptology EPrint Archive, Report 2017/511*. https://eprint.iacr.org/2017/511

Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J. B., Seurin, Y., & Vikkelsoe, C. (2007). PRESENT: An Ultra-Lightweight Block Cipher. In *Cryptographic Hardware and Embedded Systems - CHES 2007* (pp. 450–466). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-74735-2_31

Bouillaguet, C., Dunkelman, O., Leurent, G., & Fouque, P.-A. (2010). Another Look at Complementation Properties. In S. Hong & T. Iwata (Eds.), *Fast Software Encryption. FSE 2010. Lecture Notes in Computer Science, vol 6147* (pp. 347–364). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-13858-4_20

Boura, C., Canteaut, A., & De Cannière, C. (2011). Higher-order differential properties of Keccak and Luffa. In A. Joux (Ed.), *Fast Software Encryption. FSE 2011. Lecture Notes in Computer Science, vol 6733* (pp. 252–269). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-21702-9_15

Courtois, N. T., & Pieprzyk, J. (2002). Cryptanalysis of block ciphers with overdefined systems of equations. In Y. Zheng (Ed.), *Advances in Cryptology — ASIACRYPT 2002. ASIACRYPT 2002. Lecture Notes in Computer Science, vol 2501* (pp. 267–287). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-36178-2_17

Cryptographic competitions: CAESAR submissions, (2014). https://competitions.cr.yp.to/caesar-submissions.html

Daemen, J., Hoffert, S., Peeters, M., Van Assche, G., & Van Keer, R. (2019). Xoodyak, a lightweight cryptographic scheme. *In: Lightweight Cryptography Standardization Process Round 2 Submission, NIST, Special Issue 1*, 60–87.

https://doi.org/10.13154/tosc.v2020.iS1.60-87

Daemen, J., Mennink, B., & Van Assche, G. (2017). Full-state keyed duplex with built-in multi-user support. In T. Takagi & T. Peyrin (Eds.), *Advances in Cryptology – ASIACRYPT 2017. ASIACRYPT 2017. Lecture Notes in Computer Science, vol 10625* (pp. 606–637). Springer, Cham. https://doi.org/10.1007/978-3-319-70697-9_21

Daemen, J., & Rijmen, V. (2001). The Wide Trail Design Strategy. In B. Honary (Ed.), *Cryptography and Coding. Cryptography and Coding 2001. Lecture Notes in Computer Science, vol 2260* (pp. 222–238). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45325-3_20

Daemen, J., & Rijmen, V. (2002). *The Design of Rijndael - The Advanced Encryption Standard*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-04722-4

Derbez, P., Huynh, P., Lallemand, V., Naya-Plasencia, M., Perrin, L., & Schrottenloher, A. (2020). Cryptanalysis Results on Spook. In D. Micciancio & T. Ristenpart (Eds.), *Advances in Cryptology – CRYPTO 2020. CRYPTO 2020. Lecture Notes in Computer Science, vol 12172* (pp. 359–388). Springer, Cham. https://doi.org/10.1007/978-3-030-56877-1_13

Diffie, W., & Hellman, M. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, *22*(6), 644–654. https://doi.org/10.1109/TIT.1976.1055638

Dinur, I., & Shamir, A. (2009). Cube attacks on tweakable black boxp Polynomials. In A. Joux (Ed.), *Advances in Cryptology - EUROCRYPT 2009. EUROCRYPT 2009. Lecture Notes in Computer Science, vol 5479* (pp. 278–299). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-01001-9_16

Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., Mennink, B., Primas, R., & Unterluggauer, T. (2019). ISAP v2.0. *In: Lightweight Cryptography Standardization Process Round 2 Submission, NIST*. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/isap-spec-round2.pdf

Dobraunig, C., Mendel, F., Eichlseder, M., & Schläffer, M. (2019). Ascon v1.2. *In: Lightweight Cryptography Standardization Process Round 2 Submission, NIST, Light. Cryptogr. Stand. Process round 2 submission, NIST*. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/ascon-spec-round2.pdf

Goudarzi, D., & Rivain, M. (2017). How fast can higher-order masking be in software? In J. Coron & J. Nielsen (Eds.), *Advances in Cryptology – EUROCRYPT 2017. EUROCRYPT 2017. Lecture Notes in Computer Science, vol 10210* (pp. 567–597). Springer, Cham. https://doi.org/10.1007/978-3-319-56620-7_20

Grassi, L., Rechberger, C., & Rønjom, S. (2017). Subspace Trail Cryptanalysis and its Applications to AES. *IACR Transactions on Symmetric Cryptology*, *2016*(2),

192–225. https://doi.org/10.13154/tosc.v2016.i2.192-225

Gross, H., Mangard, S., & Korak, T. (2017). An Efficient Side-Channel Protected AES Implementation with Arbitrary Protection Order. In H. Handschuh (Ed.), *Topics in Cryptology – CT-RSA 2017. CT-RSA 2017. Lecture Notes in Computer Science, vol 10159* (pp. 95–112). Springer, Cham. https://doi.org/10.1007/978-3-319-52153-4_6

Grosso, V., Leurent, G., Standaert, F., Varici, K., Durvaux, F., Gaspar, L., Kerckhof, S., & Inria, E. P. I. (2014). SCREAM & iSCREAM Side-Channel Resistant Authenticated Encryption with Masking. *CEASAR Competition*, *280141*, 1–33. http://perso.uclouvain.be/fstandae/SCREAM/SCREAM_v2.pdf

Grosso, V., Leurent, G., Standaert, F. X., & Varıcı, K. (2015). LS-designs: Bitslice encryption for efficient masked software implementations. In C. Cid & C. Rechberger (Eds.), *Fast Software Encryption. FSE 2014. Lecture Notes in Computer Science, vol 8540* (pp. 18–37). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-46706-0_2

Guo, C., Pereira, O., Peters, T., & Standaert, F. X. (2019). Towards low-energy leakage-resistant authenticated encryption from the duplex sponge construction. *Cryptology EPrint Archive, Report 2019/193*. https://eprint.iacr.org/2019/193

Guo, J., Jean, J., Nikolic, I., Qiao, K., Sasaki, Y., & Sim, S. M. (2016). Invariant Subspace Attack Against Midori64 and The Resistance Criteria for S-box Designs. *IACR Transactions on Symmetric Cryptology*, *2016*(1), 33–56. https://doi.org/10.46586/tosc.v2016.i1.33-56

Guo, J., Peyrin, T., Poschmann, A., & Robshaw, M. (2011). The LED block cipher. In B. Preneel & T. Takagi (Eds.), *Cryptographic Hardware and Embedded Systems – CHES 2011. CHES 2011. Lecture Notes in Computer Science, vol 6917* (pp. 326–341). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-23951-9_22

Hell, M., Johansson, T., Meier, W., Jonathan, S., & Hell, M. (2019). Grain-128AEAD - A lightweight AEAD stream cipher. *In: Lightweight Cryptography Standardization Process Round 2 Submission, NIST*. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/grain-128aead-spec-round2.pdf

Hellman, M. (1980). A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, *26*(4), 401–406. https://doi.org/10.1109/TIT.1980.1056220

Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B. S., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., & Chee, S. (2006). HIGHT: A new block cipher suitable for low-resource device. In L. Goubin & M. Matsui (Eds.), *Cryptographic Hardware and Embedded Systems - CHES 2006. CHES 2006. Lecture Notes in Computer Science, vol 4249* (pp. 46–59). Springer, Berlin, Heidelberg. https://doi.org/10.1007/11894063_4

Huang, H. W. and T. (2019). TinyJAMBU: A Family of Lightweight Authenticated Encryption Algorithms. *In: Lightweight Cryptography Standardization Process*

Round 2 Submission, NIST. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/TinyJAMBU-spec-round2.pdf

ISO/IEC 29192-2:2019, Information technology — Security techniques — Lightweight cryptography — Part 2: Block ciphers, 2019. (2019).

Iwamoto, M., Peyrin, T., & Sasaki, Y. (2013). Limited-Birthday Distinguishers for Hash Functions. In K. Sako & P. Sarkar (Eds.), *Advances in Cryptology - ASIACRYPT 2013. ASIACRYPT 2013. Lecture Notes in Computer Science, vol 8270* (pp. 504–523). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-42045-0_26

Iwata, T., Khairallah, M., Minematsu, K., & Peyrin, T. (2019). Romulus v1.2. *In: Lightweight Cryptography Standardization Process Round 2 Submission, NIST*. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/Romulus-spec-round2.pdf

Jean, J., Nikolić, I., & Peyrin, T. (2014). Tweaks and keys for block ciphers: The TWEAKEY framework. In P. Sarkar & T. Iwata (Eds.), *Advances in Cryptology – ASIACRYPT 2014. ASIACRYPT 2014. Lecture Notes in Computer Science, vol 8874* (pp. 274–288). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-45608-8_15

Khovratovich, D., Nikolić, I., Pieprzyk, J., Sokołowski, P., & Steinfeld, R. (2015). Rotational cryptanalysis of ARX revisited. *Cryptology EPrint Archive, Report 2015/095*. https://doi.org/10.1007/978-3-662-48116-5_25

Knudsen, L. R. (1994). Truncated and higher order differentials. In B. Preneel (Ed.), *Fast Software Encryption. FSE 1994. Lecture Notes in Computer Science, vol 1008* (pp. 196–211). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-60590-8_16

Leander, G., Abdelraheem, M. A., AlKhzaimi, H., & Zenner, E. (2011). A Cryptanalysis of PRINTcipher: The Invariant Subspace Attack. In P. Rogaway (Ed.), *Advances in Cryptology – CRYPTO 2011. CRYPTO 2011. Lecture Notes in Computer Science, vol 6841* (pp. 206–221). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-22792-9_12

Leander, G., Minaud, B., & Rønjom, S. (2015). A generic approach to invariant subspace attacks: Cryptanalysis of Robin, iSCREAM and Zorro. In E. Oswald & M. Fischlin (Eds.), *Advances in Cryptology -- EUROCRYPT 2015. EUROCRYPT 2015. Lecture Notes in Computer Science, vol 9056* (pp. 254–283). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-46800-5_11

Leander, G., Tezcan, C., & Wiemer, F. (2018). Searching for subspace trails and truncated differentials. *IACR Transactions on Symmetric Cryptology*, *2018*(1), 74–100. https://doi.org/10.13154/tosc.v2018.i1.74-100

Liskov, M., Rivest, R. L., & Wagner, D. (2011). Tweakable Block Ciphers. *Journal of Cryptology*, *24*(3), 588–613. https://doi.org/10.1007/s00145-010-9073-y

McKay, K. A., Bassham, L., Turan, M. S., & Mouha, N. (2017). Report on lightweight cryptography. *National Institute of Standards and Technology, NISTIR 811*, 26. http://nvlpubs.nist.gov/nistpubs/ir/2017/NIST.IR.8114.pdf

Mendel, F., Rechberger, C., Schläffer, M., & Thomsen, S. S. (2009). The rebound attack: Cryptanalysis of reduced whirlpool and Grøstl. In O. Dunkelman (Ed.), *Fast Software Encryption. FSE 2009. Lecture Notes in Computer Science, vol 5665* (pp. 260–276). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-03317-9_16

NIST. (1998). *SKIPJACK and KEA Algorithm Specifications* (pp. 1–23). https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Algorithm-Validation-Program/documents/skipjack/skipjack.pdf

NIST. (2018). *Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process*. 1–17. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf

Peyrin, T. (2010). Improved differential attacks for ECHO and Grøstl. In T. Rabin (Ed.), *Advances in Cryptology – CRYPTO 2010. CRYPTO 2010. Lecture Notes in Computer Science, vol 6223* (pp. 370–392). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-14623-7_20

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, *21*(2), 120–126. https://doi.org/10.1145/359340.359342

Shirai, T., Shibutani, K., Akishita, T., Moriai, S., & Iwata, T. (2007). The 128-Bit Blockcipher CLEFIA (Extended Abstract). In A. Biryukov (Ed.), *Fast Software Encryption. FSE 2007. Lecture Notes in Computer Science, vol 4593* (pp. 181–195). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-74619-5_12

Tezcan, C. (2010). The improbable differential attack: Cryptanalysis of reduced round CLEFIA. In G. Gong & K. C. Gupta (Eds.), *Progress in Cryptology - INDOCRYPT 2010. INDOCRYPT 2010. Lecture Notes in Computer Science, vol 6498*. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-17401-8_15

Tezcan, C. (2014). Improbable differential attacks on Present using undisturbed bits. *Journal of Computational and Applied Mathematics, 259*(PART B), 503–511. https://doi.org/10.1016/j.cam.2013.06.023

Tezcan, C. (2020). Analysis of Ascon, DryGASCON, and Shamash Permutations. *Cryptology EPrint Archive, Report 2020/1458*. https://eprint.iacr.org/2020/1458

Tezcan, C. (2016). Truncated, Impossible, and Improbable Differential Analysis of ASCON. *2nd International Conference on Information Systems Security and Privacy*, 325–332. https://doi.org/10.5220/0005689903250332

Todo, Y. (2015). Structural evaluation by generalized integral property. In E. Oswald & M. Fischlin (Eds.), *Advances in Cryptology -- EUROCRYPT 2015.*

EUROCRYPT 2015. Lecture Notes in Computer Science, vol 9056. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-46800-5_12

Tromer, E., Osvik, D. A., & Shamir, A. (2010). Efficient cache attacks on AES, and countermeasures. *Journal of Cryptology*, *23*(1), 37–71. https://doi.org/10.1007/s00145-009-9049-y

Tsunoo, Y., Tsujihara, E., Shigeri, M., Saito, T., Suzaki, T., & Kubo, H. (2008). Impossible Differential Cryptanalysis of CLEFIA. In K. Nyberg (Ed.), *Fast Software Encryption. FSE 2008. Lecture Notes in Computer Science, vol 5086* (pp. 398–411). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-71039-4_25

Turan, M. S., McKay, K. A., Çalık, Ç., Chang, D., & Bassham, L. (2019). *Status report on the first round of the NIST lightweight cryptography standardization process*. 1–13. https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8268.pdf

Xiang, Z., Zhang, W., Bao, Z., & Lin, D. (2016). Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In J. Cheon & T. Takagi (Eds.), *Advances in Cryptology – ASIACRYPT 2016. ASIACRYPT 2016. Lecture Notes in Computer Science, vol 10031* (pp. 648–678). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-53887-6_24