MDS MATRICES OVER RINGS FOR DESIGNING LIGHTWEIGHT BLOCK CIPHER

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

BY

GÖKÇE YETİŞER

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

SEPTEMBER 2021

Approval of the thesis:

MDS MATRICES OVER RINGS FOR DESIGNING LIGHTWEIGHT BLOCK CIPHER

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

Prof. Dr. A. Sevtap Selçuk Kestel _____
Director, Graduate School of **Applied Mathematics**

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

Assoc. Prof. Dr. Oğuz Yayla _____
Supervisor, **Cryptography, METU**

Examining Committee Members:

Assoc. Prof. Dr. Murat CENK
Institute of Applied Mathematics, METU _____

Assoc. Prof. Dr. Oğuz YAYLA
Institute of Applied Mathematics, METU _____

Assoc. Prof. Dr. Sedat AKLEYLEK
Dept. of Computer Engineering, Ondokuz Mayıs University _____

Assist. Prof. Dr. Adnan ÖZSOY
Dept. of Computer Engineering, Hacettepe University _____

Assoc. Prof. Dr. Fatih SULAK
Dept. of Mathematics, Atılım University _____

Date: _____

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

Name, Last Name: GÖKÇE YETİŞER

Signature :

ABSTRACT

MDS MATRICES OVER RINGS FOR DESIGNING LIGHTWEIGHT BLOCK CIPHER

Yetişer, Gökçe
M.S., Department of Cryptography
Supervisor : Assoc. Prof. Dr. Oğuz Yayla

September 2021, 46 pages

The primary intention of this thesis is to generate lightened Maximum Distance Separable (MDS) matrices so as not to have a high implementation cost while providing solid diffusion to a cipher. In this direction, lightweight cryptography along with the current capabilities of devices and some design principles and permutation layer is given briefly. Next, the diffusion property of block cipher design is studied, where the permutation layer is generally an invertible linear transformation, and they are generally represented as matrices. In this thesis, we mainly focus on the design of MDS matrices over rings for efficient block ciphers; in particular, the matrices resulting in better implementation costs are considered. To generate MDS matrices with lower XOR gate number and depth, we advance the technique given by Li et. al. and expand the search space and reach better implementation results. For the implementation cost, we count the number of XOR gates and the depth by benefiting a modified version of Boyar's SLP heuristic algorithm, with depth awareness, given in the same article. In particular, we tabulate some matrices with good implementation costs. The best involutory MDS matrix, with branch number 5, found in this study has only 64 XOR gates with depth 3, which is a new record. The matrices found could be used in any cipher to provide diffusion, as long as the input size of the one-time multiplication operation is set to be 32 bits. In addition, it also provides a strong permutation with optimal cost.

Keywords: MDS matrices, Block ciphers, Lightweight cryptography

ÖZ

HAFİF BLOK ŞİFRELEME TASARIMI İÇİN HALKALAR ÜZERİNDE MDS MATRİSLERİ

Yetişer, Gökçe
Yüksek Lisans, Kriptografi Bölümü
Tez Yöneticisi : Doç. Dr. Oğuz Yayla

Eylül 2021, 46 sayfa

Bu tezin ana amacı, kriptografik bir algoritmaya iyi yayılım sağlarken uygulama maliyetini düşürmek için hafifletilmiş MDS matrisleri oluşturmaktır. Bu doğrultuda, hafif algoritmalarda kullanılan cihazların mevcut kapasiteleri ve bazı tasarım ilkeleri ile birlikte permütasyon katmanı tanıtılmaktadır. Daha sonra, permütasyon katmanının yayılım özelliği çalışılmıştır. Permutasyon katmanı genellikle tersinebilir doğrusal bir dönüşümdür ve matrislerle gösterilir. Bu tezde, etkili blok şifreleri tasarlamak için esas olarak halkalar üzerinde MDS matrisler üzerinde duruyoruz; örneğin, daha az uygulama maliyetleri ile sonuçlanan matrisleri inceliyoruz. Daha düşük XOR kapı sayısı ve derinliği olan MDS matrisleri oluşturmak için, arama uzayını genişleterek Li ve diğerlerinin makalesinde kullanılan tekniği geliştirdik ve daha iyi uygulama sonuçlarına ulaştık. Uygulama maliyeti için, aynı makalede verilen Boyar'ın SLP algoritmasının derinlik kavramı göz önüne alınarak değiştirilmiş bir versiyonundan yararlanarak XOR kapılarının sayısını ve derinliğini hesapladık. Özellikle, iyi uygulama maliyetleri olan bazı matrisleri tablo haline getirdik. Bu çalışmada bulunan 5 dallanma sayılı en iyi tersi kendisine eşit olan MDS matrisi, 64 XOR kapısıyla ve 3 derinliğiyle hesaplanabilmektedir ve bu şimdiye kadar aynı boyuttaki matrisler arasında ulaşılan en iyi uygulama maliyetidir. Bulunan bu matrisler, tek seferde çarpma işleminin girdi boyutu 32 bit olara ayarlandığı sürece, her kriptografik algoritmada permutasyon sağlamak için kullanılabilir. Buna ek olarak difüzyonun güçlü olmasını

kesinleştirirken, optimal uygulama maliyeti sağlar.

Anahtar Kelimeler: MDS matrisler, Blok şifreleme, Hafifsiklet kriptografi

dedicated to the people who believe in mermaids

ACKNOWLEDGMENTS

I would like to express my very great appreciation to my thesis supervisor Assoc. Prof. Dr. Oğuz Yayla for his patient guidance, enthusiastic encouragement and valuable advice during the development and preparation of this thesis. His willingness to give his time and to share his experiences has brightened my path. By showing that he is there for me as I push my limits, it has allowed me to do more than I thought I could do. I learned a lot in many areas than I can imagine. If it weren't for him, I wouldn't have gotten this far.

I would also like to thank my examining committee members Assoc. Prof. Dr. Murat Cenk, Assoc. Prof. Dr. Sedat Akleylek, Assist. Prof. Dr. Adnan Özsoy and Assoc. Prof. Dr. Fatih Sulak for their time they spared for me and valuable comments.

I owe a lot to my friends and family who have never lost their support from me. Thanks to them, this process wore me out much less than it could have.

I would also like to thank my colleagues for their understanding and support they have always shown.

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The aim of this thesis is to create a sense of the method of generating lightweight maximum distance separable (MDS) matrices while introducing their limitations. However, even though it is not the primary intention to investigate lightweight algorithms, this thesis aims to create a general understanding of the concept of lightweightness and summarize up-to-date limitations in order to explicate the importance of the permutation layer and MDS matrices.

Lightweight cryptography is an important field of research, applied when there is restricted memory space and processing power, which has emerged over the last years with the development of resource-limited devices. The main goal of lightweight cryptography is to establish cryptographic solutions for applications in resource-constrained environments. A lightweight design is optimized concerning specific lightweight metrics. Chip area, latency, throughput, code size, power or energy consumption are included in such metrics. Some of these metrics are explained further in Chapter 2.

Some of the most popular recent lightweight block ciphers are PRESENT [16], CLEFIA [47], KATAN/KTANTAN [23], PICCOLO [46], PRINCE [17], SIMON/ SPECK [14], LEA [31], PRIDE [6], MIDORI [11], and TEA [51], where Grain v1 [30] and Trivium [24] are examples of lightweight stream ciphers. This could be deduced from the fact that MDS matrices are not only widely used in SPN ciphers but also in Feistel ciphers, in hash functions and even in stream ciphers, that the development of MDS matrices will yield benefit in a wide area of cryptography, not a barren area.

Some common design principles of lightweight algorithms are to use relatively small block-size, relatively small key-size and many simple rounds. These standard features cause some weaknesses in them. The restrictions make it very difficult to optimize the existing algorithms to the requirements of lightweight cryptography since many of them would lose much resistance, being limited in computing resources. While generating an algorithm with good reliability and performance would exceed the limited area of lightweight devices, the capacity of a low-cost algorithm would be limited.

Lightweight algorithms' diffusion layers usually consist of simple bit permutations. These simple permutations do not contribute as much to security and diffusion as desired. In addition, the number of rounds must be increased for this permutation, which creates a superficial layer to contribute round simplicity and provide sufficient diffusion. Increasing the number of rounds also increases latency, as explained in detail later in Chapter 2. Consequently, the permutation layer of lightweight algorithms is improved by producing functions with better diffusion property whose cost is also optimised while the cost is stabilized.

MDS matrices offer certain diffusion properties. Moreover, when MDS matrices are compared with matrices of the same size, it appears that MDS matrices provide the best diffusion. Many modern ciphers use MDS matrices to ensure diffusion. Advanced Encryption Standard [22], Camellia [8], Twofish [42, 43], MUGI (stream cipher) [50], SHARK [38] and Square [21] are the cryptographic ciphers that benefit from MDS matrices. MDS matrices are also used in the design of hash functions. Hash functions like Maelstrom [26], Grostl [27], Whirlpool [13] and PHOTON family lightweight hash functions [28] use MDS matrices as main part of their diffusion layers.

MDS matrices not only have good diffusion properties but also the diffusion rate of these matrices can be easily measured using branch number and fixed point number (FPN). The branch number is used to measure the diffusion ratio and gives a method to define security. MDS matrices are, in fact, the optimal diffusion matrices as the diffusion layer need to create a dependency between input and output words, with a high branch number and MDS matrices have the highest branch number a matrix can have. FPN, which measures the fixed points, is another criterion to determine the

quality of diffusion. Linear transformations which have high branch number and low FPN are desirable to sustain great diffusion. Some MDS matrix examples with their branch number and FPN is given in Table 1.1.

Table 1.1: Famous MDS Matrices

	matrix	**branch number**	**FPN**
AES	4 x 4 MDS matrix in $GF(2^8)$	5	2^8
KHAZAD	8 x 8 involutory MDS matrix in $GF(2^8)$	9	2^{32}
CAMELLIA	8 x 8 MDBL matrix in $GF(2^8)$	5	2^8
ARIA	16 x 16 involutory MDBL matrix in $GF(2^8)$	8	2^{72}

In addition to the diffusion characteristics, the implementation cost is also essential, especially while considering lightweight algorithms. Implementation cost is measured using some metrics such as distance, direct XOR count, XOR gate number and depth. Considering the implementation costs, lower XOR gate numbers and small depth are tried to achieve, as matrices with lower XOR gate numbers are lighter, and those with low depth will have less latency.

MDS matrices are generally matrices with high implementation costs depending on the diffusion properties they have. MDS matrices would be a great solution to the weak security level caused by superficial layers in lightweight algorithms by reducing the implementation cost. In this thesis, the implementation cost is considered while generating the MDS matrix, and the constructed matrices give the best XOR gate number and depth so far, see Appendix A.

The involutory condition is another feature that plays a vital role in the lightness of a matrix. Involutory matrices are the matrices that are their own inverse. Implementing an involutory MDS matrix removes the time difference between encryption and decryption processes, as both use the same matrix. This feature not only benefits us in cryptanalysis, especially in timing attacks but also supports the desire to provide low implementation costs by using the same operations used during encryption and decryption. The involutory feature also ensures that just storing one matrix is sufficient and thus takes up little memory space. Further information related to the metrics will be given in Chapter 3.

Linear transformations used in block ciphers provide diffusion by linearly mixing bits

of fixed-size input block to produce the corresponding output block of the same size. Therefore, they affect the cipher's security directly. In the literature, the constructions of these transformations are various. While some of them are based on algebraic construction, others are based on random-like construction. There are search-based, direct generation, and generation methods using custom matrices. In this study, subfield construction method have been used to obtain lightweight involutory MDS matrices by considering the restrictions and criteria described in Chapter 2. We improved the method applied in the article named "Constructing Low-latency Involutory MDS Matrices with Lightweight Circuit" [36] and performed an MDS matrix search. As a result, we have reached different and lower cost matrices by expanding the search space, which was narrowed by the restrictions stated later. We benefit from division operation on Laurent polynomial ring to avoid missing any solution among the criteria we have determined. The method and reached matrices are further mentioned in Chapter 3.

1.1 Organization

This thesis aims to generate lightened Maximum Distance Separable (MDS) matrices used in lightweight cryptography. Firstly, we study lightweight cryptography with its limits. The design principles of lightweight block ciphers and their permutation layer are given in Chapter 2. While providing information on lightweight cryptography, we focus on its requirements and why it is needed. Then, after examining the requirements and the limits of lightweight devices and lightweight cryptography, the current applications are presented. In addition, the performances of symmetric algorithms are also examined briefly. Because this thesis believes it is important to see current algorithms and benchmarking results to see why and what for, it is desired to lighten the permutation layer and MDS matrices while keeping its security at the top. As an example of design principles of lightweight algorithms the area requirements in hardware implementations and the latency were introduced as they are significant factors to pay attention to. In Chapter 3, a known MDS matrix generation method has been given. The best results obtained are given in Appendix A. Most lightweight algorithms use smaller S-boxes; thus, they need smaller matrices in their permutation

layers. Although most lightweight algorithms do not even use complex operations, such as matrix multiplication, to slow the algorithm down in their permutation layers, this thesis believes it would be beneficial to use suitable matrices like MDS matrices, if they could lighten and expedite. There is another weakness of the matrix found in this thesis, that is, the size of the matrix. Despite this weakness, the method used could be used to create smaller matrices. Another option could be changing the field studied in order to make the matrices more desirable.

CHAPTER 2

LIGHTWEIGHT CRYPTOGRAPHY AND PERMUTATION LAYER

This thesis is mainly focused on lightweight block ciphers and strengthening the permutation layer by lowering the implementation cost of permutation-strong MDS matrices. Therefore, this chapter emphasises general features, limits, and design criteria of lightweight block ciphers related to permutation layers of block ciphers and MDS matrices.

2.1 Lightweight Cryptography

Lightweight cryptography is an encryption method applied when there is restricted memory space and processing power. Lightweight cryptographic algorithms provide security for critical information as it appears in every part of daily life with the development of technology. In addition, along with the widespread use of lightweight devices, the need to increase the security levels of lightweight algorithms has become more visible.

A lightweight block cipher is a type of secret key cryptography that is used to create devices with limited resources. Some common design principles of lightweight algorithms are to use relatively small block-size, relatively small key-size, and many simple rounds. However, these standard features cause some weaknesses in them. Moreover, due to the constraints, optimizing current methods to meet the needs of lightweight cryptography is extremely challenging. While generating an algorithm with good reliability and performance would exceed the limited area, the capacity

of a low-cost algorithm would be limited. When designing a cryptographic algorithm dedicated to a resource-limited device, the main goal should be to lighten the algorithm in every possible aspect, such as memory usage, chip size, power consumption. However, arrangements were made to lighten and accelerate their lack of adequate security. Despite the fact that most cryptographic solutions focus on high-level security, restricted resources make it difficult to properly implement a ample cryptographic functions on resource-constrained devices. Therefore, when designing a lightweight algorithm, a trade-off must be made between security, throughput and area. To generate secure and low-cost cryptographic algorithms, the lightened layers that cause security compromises should be re-examined.

The most popular design methods for creating a block cipher algorithm are the Feistel network and substitution permutation network (SPN). Between these two types, the traditional Feistel ciphers take much time and consume a large amount of energy as they need far more round functions than the substitution permutation networks. A substitution permutation cipher defines a particular structure of the round functions in a product cipher. In substitution permutation networks, linear and nonlinear parts are applied in two separate layers during encryption. Thereby, the rounds consist of applying a nonlinear operation, substitution layer, and the application of a linear transformation, permutation layer. The linear layer, the permutation layer, precisely the main objective of this thesis, can be defined by a linear transformation. Then, the application of permutation corresponds to matrix multiplication. The permutation layer should diffuse the information over the whole state. In particular, in this study, the diffusion layer has been optimised so that it can be used in a lightweight application while providing the diffusion feature in the best way.

The optimization of a lightweight design is done concerning specific lightweight metrics. Such metrics include chip area, latency, throughput, code size, power or energy consumption. Chip area and latency are briefly explained.

Area in Hardware [15]. The required hardware area is commonly measured in Gate Equivalents (GE), and one GE determines the needed area for implementing a single two-input bitwise NAND gate. Present, requiring 1570 GE, is one of the first examples designed to reduce the area. Present [16], benefits from simple bit permutations

in its permutation layer as bit permutations do not require any gate.

When the permutation layer is designed more complexly, XOR gates are needed in its implementation. Hence, for the lightness of the implementation, it is significant to reduce the number of xor gates.

Latency [15]. The time takes to encrypting and providing a message is called latency. Especially in applications that require fast response, low-latency algorithms are preferred. One of the first low-latency block ciphers is Prince [17]. Prince, who has only a small number of rounds, is not implemented in a round-based manner. It takes many iterations of rounds to create a secure algorithm. Despite this, iteration of many rounds would significantly increase the latency of the cipher.

Thus, while designing a lightweight cryptographic algorithm, the best trade-off between the lightweightness of the rounds and the number of rounds are aimed to be applied.

2.1.1 Lightweight Devices and Their Limitations

The goal of lightweight cryptography is to create a security solution that can function on devices with limited resources by using less memory, computational resources, and power supply. There is a need for expanding the applications of cryptography to constrained devices.

The leading software and hardware performance benchmarking initiatives considered during the lightweight cryptography standardization process are summarized further in this section. The limitations are significant to understand the lightweight implementations.

2.1.1.1 Software Benchmarking

The performance of software on microcontrollers is a critical factor for evaluating cryptographic algorithms. The NIST report reviewed many benchmarking initiatives that span a wide variety of target platforms, from 8-bit microcontrollers with low

memory through 32-bit and 64-bit microcontrollers [48]. Table 2.1 summarizes the specifications of the microcontrollers utilized in the benchmarking projects.

Table 2.1: Micro-controllers in some previous work

Work	Microcontroller	Processor	Word size	Clock speed	Flash	RAM
NIST	ATmega328P	AVR	8-bit	16 MHz	32 KB	2 KB
	ATmega4809	AVR	8-bit	20 MHz	48 KB	6 KB
	SAM D21	ARM Cortex-M0+	32-bit	48 MHz	256 KB	32 KB
	nRF52840	ARM Cortex-M4	32-bit	64 MHz	1 MB	256 KB
	PIC32MX340F512H	MIPS32 M4K	32-bit	80 MHz	512 KB	32 KB
	ESP8266	Tensilica L106	32-bit	80 MHz	4 MB	80 KB
Renner et al. [39]	ATmega328P	AVR	8-bit	16 MHz	32 KB	2 KB
	STM32F103	ARM Cortex-M3	32-bit	72 MHz	64 KB	20 KB
	STM32F746ZG	ARM Cortex-M7	32-bit	216 MHz	1 MB	320 KB
	ESP32 WROOM	Tensilica Xtensa LX6	32-bit	240 MHz	4 MB	520 KB
	Kendryte K210	RISC-V (Dual Core)	64-bit	400 MHz	16 MB	8 MB
Weatherley [4]	ATmega2560	AVR	8-bit	16 MHz	256 KB	8 KB
	AT91SAM3X8E	ARM Cortex-M3	32-bit	84 MHz	512 KB	96 KB
	ESP32	Tencilica Xtensa LX6	32-bit	240 MHz	4 MB	520 KB

Summary of the specifications of microcontrollers, from 8-bit limited memory to 32-bit and 64-bit, which is evaluated in the NIST's most recent lightweight report is given in Table 2.1. As can be seen, the RAM varies between 2 KB to 8MB, while the flask capacity is in the range of 32KB to 16 MB. Moreover, the lowest clock speed is 16 MHZ, where the highest one is 400 MHZ.

2.1.1.2 Hardware Benchmarking

NIST evaluated results from hardware benchmarking initiatives. In the given result, the previously mentioned quantum-resistant algorithms were taken into account.

The Cryptographic Engineering Research Group at George Mason University [37] released performance data on three distinct systems. The results are summarized for the algorithms that support quantum resistance.

ASCON, Gimli, KNOT, DryGASCON, Spook, and TinyJAMBU were the top AEAD performers on the **Xilinx Artix-7** platform in terms of throughput. TinyJAMBU and Gimli both have implementations that were less than 1000 Look Up Tables in size (LUTs). SHA-2 was outperformed by Gimli and ASCON when it came to hashing. The smallest implementations that support hashing are Gimli and SATURNIN. The only implementation that went above the resource limit was SPARKLE.

The **Intel Cyclone 10 LP FPGA** implementations on the Xilinx Artix-7 platform produced comparable results. ASCON, Gimli, KNOT, DryGASCON, TinyJAMBU, Spook, and SATURNIN all outperformed AES-GCM in terms of throughput, with the AES-GCM implementation exceeding the 5000 LE budget. TinyJAMBU has the smallest implementation through considered algorithms. Gimli also demonstrated faster hashing than SHA-2. The candidate with the smallest implementations supporting hashing was Gimli.

Results were also similar on the **Lattice ECP5**, where Gimli, ASCON, KNOT, DryGASCON, and TinyJAMBU implementations processed plaintext faster than AES-GCM; and Gimli exhibited higher hashing throughput than SHA-2. Gimli was the candidate with the smallest implementations supporting hashing.

According to Khairallah et al. [33], TinyJAMBU had the smallest implementation and firm performance when optimized for the low area. When the area is limited to 9000 GE, again, TinyJAMBU displays the best results.

Five distinct ASIC cell libraries and two synthesis tools are available to Aagaard and Zidaric [5]. The tiniest footprint was TinyJAMBU. However, It has a poor performance. Gimli and ASCON were the choices with the best energy-efficient implementations. Moreover, TinyJAMBU was also very energy efficient at lower throughputs. TinyJAMBU excels once again when considering area $\times$ energy. ASCON had low area $\times$ energy with good throughput.

The platforms and metrics that were used in these initiatives are listed in the Table 2.2 below.

Table 2.2: Hardware benchmarking

Work	Platforms	Metrics
GMU CERG Group [37]	Xilinx Artix-7 Intel Cyclone 10 LP Lattice Semiconductor ECP5	Resource utilization (LUT or LE, flip-flops) Maximum clock frequancy (MHz) Throughputs (Mbits/s) Energy per bit (nJ/bit)
Khairallah et al. [33]	TSMC 65nm FDSOI 28 nm	Area (μm^2 and GE) Clock period (ns) Power (mW) Energy (mJ)
Aagaard and Zidaric [5]	ST Micro 65nm TSMC 65nm ST MİCro 90nm TSMC 90nm ARM/IBM 130nm	Throughputs (bits per cycle) Area (GE) Energy (nJ) Area $\times$ Energy (GE $\times$ nJ) Clock speed (GHz)

An idea about the limitation of the devices used is formed. In order to see the needs in this area, an idea about the implementation values of the algorithms also needs to be formed. In NIST's report, the implementations of current lightweight algorithms are compared with AES and SHA in terms of code size and time. Lightweight symmetric algorithms generally have similar code sizes to AES and SHA. There are algorithms with smaller and larger code sizes, but neither difference is very noticeable. If one compares asymmetric algorithms, although there are algorithms that can meet the limits of lightweight devices, most of the asymmetric algorithms, particularly quantum-resistant ones, exceed the limits. On the other hand, although the current microcontroller resources are limited, these limits are growing with technology development.

Suppose the run times of the algorithms are examined. In that case, it can be seen that while the symmetric algorithms generally give results close to AES and SHA and can work in microseconds, asymmetric algorithms run much slower.

2.2 Permutation Layer

The diffusion layer is a cryptographically important linear function that satisfies permutation, and it plays a vital role in providing resistance against many attacks on block ciphers. The diffusion could be satisfied with bit permutations, shuffles or some methods based on linear algebra such as multi-permutations and MDS matrices.

The branch number is a method to measure the diffusion rate on permutation layers. The branch number of a diffusion layer is defined as the minimum sum of the nonzero input and output bytes. The diffusion layer spreads the nonzero bits or nibbles that increase the number of active S-boxes in a differential/linear trace. Thus the maximum effect of the substitution layer is obtained. A high branch means that changing a single byte of the input will change the output a lot, which would be expected on an excellent diffusion layer. The maximum branch number of a linear transformation is satisfied with MDS matrices.

However, lightweight algorithms do not frequently prefer linear transformations with superior diffusion properties due to their weight. The inability to use superior diffusion functions due to their weight is precisely the problem this thesis deals with. A maximum distance separable (MDS) matrix represents a function with specific diffusion properties. Such matrices are the best examples of the heavily loaded linear transformations with superior diffusion property mentioned. By reducing the implementation cost, MDS matrices would be a great solution to the weak security level caused by superficial permutation layers in lightweight algorithms.

2.2.1 Maximum Distance Separable Matrices

A maximum distance separable (MDS) matrix is a matrix representing a function with specific diffusion properties. Such matrices are the best examples of the heavily loaded linear transformations with superior diffusion property mentioned earlier. Although the cost of MDS matrices makes them unusable in lightweight cryptography, by reducing the implementation cost, MDS matrices would be a great solution to the weak security level caused by superficial layers in lightweight algorithms.

MDS matrices emerged in 1994 with the introduction of multi-permutations as the formalization of diffusion layer by Schnorr [44], and Vaudenay [49]. After that, in 1995, Vaudney [32] showed the usefulness of multi-permutation in the design of cryptographic primitives. From 1994 to 1996, Heys and Tavares [7] showed that the replacement of the permutation layer of Substitution Permutation Networks (SPNs) with a diffusive linear transformation increases the cipher's resistance to differential and linear cryptanalysis by improving the avalanche characteristics of the block ci-

pher. Then, MDS matrices started to become popular when Rijmen et. al. [40] first used these matrices in a cryptographic algorithm, SHARK. Later such matrices were used in many ciphers. Now the usefulness of MDS matrices in the diffusion layer is well understood.

Although low Hamming weight is desirable in the generation of lightweight MDS matrices, the actual point needed to be considered is the minimisation of the cost of the multiplication. Thus the implementation properties of matrices needed to be considered while generating an MDS matrix.

There are two ways of constructing MDS matrices one can start with a known MDS code or search for matrices that satisfy the non-singular sub-matrix condition. The direct construction methods could be based on Cauchy matrices [52], companion matrices or Vandermonde matrices [35], where search based methods usually benefit from recursive or hybrid structures of a special form of a matrix. Such particular matrix forms include circulant and finite field Hadamard matrices. One of the well-known MDMS matrices AES's MicColumn matrix is a circulant matrix. Circulant matrices are preferred as each row differs from the previous row by a shift so that they can be easily implemented in hardware. On the other hand, Hadamard matrices are helpful in constructing involutory MDS matrices. Involutory diffusion layers have an essential effect on the performance, as they provide the same implementation properties for both encryption and decryption.

The thesis is more about generating lightweight involutory MDS matrices on block ciphers. However, MDS matrices are not only used in block ciphers but also used in some hash functions or even stream ciphers. AES, KHAZAD and CAMELIA are block cipher examples that use MDS matrices in their design. In addition, hash functions like Whirlpool, PHOTON and Whirlwind use MDS matrices. MUGI, which is a stream cipher algorithm, also uses an MDS matrix [41].

2.3 Conclusion

In the first part of this thesis, the concept of lightweightness and some design principles are briefly mentioned to explain the importance of the low-cost permutation layer

and the maximum distance separable matrices. In order to have a general understanding of lightweight cryptography, it is essential to see the limitations and capabilities of the devices used. Thus, specifications of some devices used are also given very briefly in this chapter.

Furthermore, considerations are stated for designing a lightweight algorithm. The main focus of this thesis is to strengthen the permutation layers of lightweight block ciphers by allowing the permutationally vital functions to be used by reducing the cost.

The rest of the thesis deals with symmetric algorithms and their permutation layers. When studying the permutation layer, lightweight requirements were taken into account to affect the weight of the algorithm minimally.

CHAPTER 3

MDS MATRICES OVER RINGS FOR DESIGNING BLOCK CIPHERS

A block cipher is a mapping from a plain-text space into a cipher-text space, satisfying rigid confusion and diffusion properties. To achieve confusion and diffusion, Shannon [45] suggested the use of two separate functions in a block cipher [10]. The permutation layer mainly provides diffusion, whereas the substitution layer provides confusion.

The diffusion in the permutation layer is satisfied with linear operations such as bit permutations, shuffles and linear transformations. Maximum distance separable (MDS) matrices, an example of linear transformations, are commonly used in cryptographic algorithms.

MDS matrices have good diffusion properties. Along with specific diffusion properties, these matrices provide security against both linear and differential cryptanalysis. They are used not only in the permutation layer but also sometimes in the substitution layer.

Although MDS matrices have good diffusion properties, they have high implementation costs. Thus, finding MDS matrices having good implementation properties is essential. In recent years, there has been much work on the construction of MDS matrices with a low implementation cost in the context of lightweight cryptography. For instance, the number of XOR gates, a frequently studied area when trying to reduce application load, is a significant point to get attention on with the depth of the circuit. They are used to evaluate the matrix's implementation properties. Despite its

appealing nature, finding such matrices is not an easy task. In this article, the method named sub-field construction is used to generate MDS matrices. It has been preferred to use the sub-field construction method because this method allows us to have the same diffusion properties while lowering the number of required operations.

With the development of technology, lightweight devices have become widespread, and with this increase, lightweight algorithms have been used more frequently. Therefore, much research has been conducted on lightweight cryptographic algorithms and strengthening them. Some points to be considered to produce a robust lightweight algorithm were explained in the previous section. Along with these points, this thesis focuses on strengthening the permutation layer by working on lightweight MDS matrices to strengthen such algorithms. This thesis is not the first to work on the lightweight MDS matrices. However, new results have been achieved even though the matrix size studied is relatively large. Section, 3.1 mentions similar studies and the results achieved.

3.1 Previous Work

In the context of lightweight cryptography, many studies have been done on the construction of MDS matrices with low implementation cost.

Most of the implementation cost is due to matrix multiplication in the linear layer. Thus, significant work has gone into lowering the cost of implementing MDS matrices, which are excellent diffusion providers. Some studies have been done to reduce the cost of a known matrix, such as the MixColumn matrix of AES, while others searched for new matrices that allow optimal implementation by design. The first approach is used to improve the efficiency of a standardized cipher's implementation. The second, on the other hand, may lead to new ciphers with improved implementation characteristics.

The optimization can be done in two ways. These are local optimization and global optimization. The majority of prior attempts have focused on local optimization by constructing MDS matrices with computationally efficient coefficients. Notably, this resulted in a matrix, $\in M_4(M_8(\mathbb{F}_{\mathbf{2}}))$, with a direct XOR number of only 106, while

the direct implementation of AES's MixColumn matrix required 152-bit XOR. More recently, techniques based on global optimization, which deal with the reuse of some intermediate variables, has been introduced. Kranz et al. [34], in particular, employed optimization techniques to obtain a suitable implementation from an MDS matrix description. The cost of constructing the MixColumn matrix was reduced to 97 bitwise XOR and they presented a new matrix that had only 72 bitwise XOR operations. Besides, the study [25] using global optimization in a different way found optimum circuits generating MDS matrices by searching across a space of circuits. They reached an MDS matrix with only 67 bitwise XOR.

Guo, Peyrin and Poschmann [28] introduced another significant idea to reduce the implementation cost of MDS matrices in the lightweight hash function PHOTON. After that, they used an MDS matrix again in the lightweight block cipher LED [29]. They proposed designing an MDS matrix such that it is some integer power of a matrix that can be implemented efficiently. This allows fastening the implementation by implementing the lighter base matrix and iterate it instead of implementing the matrix directly.

Duval and Leurent [25] revealed 32×32 binary MDS matrices with branch number 5. The matrix they identified can be constructed with just 67 XOR gates, where the previously best known of the same size were 72 XOR gates [34]. However, if we take depth into account, matrices would have more XOR gates. Li et.al. in [36] have stated that they have generated 32×32 binary MDS matrices with only **78 XOR gates** with **depth 4**, where the previously known lightest one with the same size has 84 XOR gates with the same depth. In the article [36], 32×32 binary MDS matrices whose implementation costs **88 XOR gates** with **depth 3** have been generated. In this thesis **better results** than [36] **has been obtained**.

3.2 Organization

In this chapter, MDS matrices with good diffusion properties and low implementation costs are considered. The research is done by expanding the study done in the article [36]. This chapter is organized as follows. First, some fundamental algebraic

structures and some additional metrics are introduced in Section 3.3. Then after mentioning the previous research, methods using the generation of MDS matrices and the results are given.

3.3 Preliminaries

We use some algebraic structures in this chapter. Their definitions and some related results are given below.

In order to determine variables in Section 3.5, we perform the division operation on the Laurent Polynomial Ring.

Definition 3.1. *A **Laurent polynomial** with coefficients in the field $\mathbb{F}$ is an algebraic object that is typically expressed in the form;*

$$\cdots + a_{-n}t^{-n} + a_{-(n-1)}t^{-(n-1)} + \cdots + a_{-1}t^{-1} + a_0 + a_1t + \cdots + a_nt^n + \ldots,$$

where the a_i are elements of $\mathbb{F}$, and only finitely many of the a_i are nonzero. Laurent polynomial ring is treated as a polynomial ring except that the variable t can also have negative powers.

Let $\mathbb{R}$ be an arbitrary ring, and $M_k(\mathbb{R})$ be the set of all $k \times k$ matrices whose entries are from $\mathbb{R}$. $M_k(\mathbb{F}_{\mathbf{2^n}})$ denotes the set of all $k \times k$ matrices over the finite field $\mathbb{F}_{2^n}$ of 2^n elements, and $M_k(GL(n, \mathbb{F}_{\mathbf{2}}))$ is the set of all $k \times k$ matrices whose elements are taken from the general linear group $GL(n, \mathbb{F}_2)$, where the general linear group is formed by all invertible $n \times n$ matrices over $\mathbb{F}_2$. Every matrix A in $M_k(\mathbb{F}_{\mathbf{2^n}})$ or $M_k(GL(n, \mathbb{F}_{\mathbf{2}}))$ can be represented as an $nk \times nk$ binary matrix. I_n is used as the $n \times n$ identity matrix and O_n is used to denote the $n \times n$ zero matrix over $\mathbb{F}_2$. Moreover, $M_k(M_n(\mathbb{F}_{\mathbf{2}}))$ denotes the set of all $k \times k$ matrices whose entries are $n \times n$ matrices with entries $\mathbb{F}_2$.

Definition 3.2. *The **characteristic polynomial** f of a binary matrix $A \in M_m(\mathbb{F}_{\mathbf{2}})$ is defined as $f(x) = \det(xI + A) \in \mathbb{F}_{\mathbf{2}}[x]$.*

Lemma 3.3. *If f is a characteristic polynomial of $A \in M_m(\mathbb{F}_{\mathbf{2}})$, then $f(A) = 0$.*

Definition 3.4. *Let $A \in M_m(\mathbb{F}_\mathbf{2})$, $f \in \mathbb{F}_\mathbf{2}[x]$ is the **minimal polynomial** of A if and only if $f(A) = 0$, and for any $g \in \mathbb{F}_\mathbf{2}[x]$ such that $g(A) = 0$, $deg(f) \leq deg(g)$.*

Note that the minimal polynomial of $A \in M_m(\mathbb{F}_\mathbf{2})$ can be reducible.

Definition 3.5. *The **companion matrix** of $f = x^m + a_{m-1}x^{m-1} + \cdots + a_1x + a_0 \in \mathbb{F}_\mathbf{2}[x]$ is defined as $m \times m$ matrix*

$$\begin{bmatrix} 0 & & & & a_0 \\ 1 & 0 & & & a_1 \\ & 1 & \ddots & & \vdots \\ & & \ddots & 0 & a_{m-2} \\ & & & 1 & a_{m-1} \end{bmatrix}.$$

It is to see that the characteristic polynomial of f's companion matrix is f.

3.3.1 Metrics

In block cipher design, a linear transformation is commonly used to ensure diffusion. The diffusion rate of a linear transformation can be easily measured using some metrics such as branch number and fixed point number. In addition to its diffusion characteristic, its implementation cost is a critical metric measured using some metrics such as distance, direct XOR count, XOR gate number, depth, shortest linear programming bound.

Definition 3.6. *$w(x)$ is known as the **weight** of x, which is defined as the number of nonzero elements in x.*

Definition 3.7. *The **branch number** of an invertible mapping is the minimum number of nonzero components in the input vector and output vector.*

Branch number helps to define the diffusion rate and gives a method to measure security against linear, and differential cryptanalysis [53]. Branch number $\beta(A)$ of an $n \times n$ matrix A, whose entries are from $\mathbb{F}_{2^m}$, can be computed as

$$\beta(A) = min\{w(x) + w(A.x^T) | x \in (\{0,1\}^m)^n, x \neq 0\}.$$

If transformation maps the input to output the same as the input, a fixed point happens in a linear transformation. In other words, say A is a linear transformation and say $A(x) = x$ then x is a fixed point.

Definition 3.8. ***Fixed point number (FPN)*** *is the number of fixed points after applying the permutation.*

For a linear transformation A, a fixed point number can be computed as

$$\text{FPN}_A = 2^{m(rank(A)-rank(A-I))},$$

where m represents the extension degree of the studied finite field $\mathbb{F}_{2^m}$.

Linear transformations with high branch number and low FPN are used for block cipher design [36].

Along with the diffusion properties of a matrix, its implementation cost is also essential. The number of operations in the linear transformation applied via a matrix depends on its weight. Its distance vector also affects memory allocation.

Definition 3.9. ***Distance*** *of a matrix is the array representing number of 1's in each row.*

The direct XOR count of a matrix resolved from its distance is directly related to the necessary number of addition operations. Note that there are one less XOR operation from the number of 1's at each row.

Definition 3.10. ***Direct XOR count*** *(DXC) is the number of 1's in the matrix minus the number of rows in the matrix. In other words, DXC(A) =* $w(A) - nk$ *for* $A \in M_{nk}(\mathbb{F}_2)$.

Although direct XOR count is easier to calculate and the upper bound for the XOR gate number, XOR gate number gives the actual number of addition.

Definition 3.11. ***XOR gate number*** *is the number of XOR gates in the implementation of an algorithm.*

The direct XOR count gives an upper bound for the XOR gate number.

Definition 3.12 ([36]). *The number of XOR gates involved on the path with the most XOR gates is called the* ***depth****.*

In this thesis, the implementation of a matrix is checked with respect to the cost of the matrix multiplication.

Definition 3.13 ([18]). ***SLP: Shortest Linear Program*** *is an algorithm, who tries to find a straight-line to compute the operation, of optimizing the cost of the implementation.*

One can obtain an estimation of a given matrix's, say $A \in M_{nk}(\mathbb{F}_2)$, cost by finding an excellent linear straight-line program corresponding to A, and this metric is denoted as SLP(A). In order to calculate XOR gate number and depth easier, the shortest linear program is used.

There are some algorithms to calculate SLP. In this thesis, a modified version of Boyar's SLP algorithm is used to give the optimal result. Boyar-Peralta algorithm starts by initializing a distance vector. Then the following steps are applied in a loop. Distance array is updated by choosing a new base element by adding two existing base elements that minimize the distance array. The process is repeated until all elements of the distance array is equal to zero. If there is a tie between two choices of the new base element, then the base element that maximizes the Euclidean norm is chosen to be the new base element. AES mix column matrix has been implemented by using this algorithm with 96 XOR gates.

Paar's algorithm is another well-known algorithm to calculate SLP. Paar's algorithm does not benefit from distance vectors. Every stage finds the pair of operands that appear most frequently in the set of equations and replaces them with a new variable. The process continues until all operands appear precisely once. It directly chooses the most frequently used sum as a new variable. The process continues until all operands appear precisely once. The algorithm, however, returns solutions that are known to be not always optimal. Nevertheless, it is always helpful to use this algorithm to decrease the gate count of larger matrices. The Boyar-Peralta method is unable to return a solution in a reasonable time.

Definition 3.14. ***Latency*** *is a measure of the time required for a sub-system or a*

component in that sub-system to process a single storage transaction or data request.

Latency is another important metric, which depends on its depth since depth affects the weight of simultaneously done operations [20].

Parallel computing is the simultaneous use of multiple computes resources to solve a computational problem. Thus, while the decrease in depth directly affects latency in parallel implementations as depth affects the weight of simultaneously done operations, it does not apply in serial implementations. In serial implementations, the number of xor gates stands out as the criterion of lightweightness.

In order to calculate the XOR gate number and depth, the shortest linear programs are used. Matrices with lower XOR gate numbers would be lighter, and those with low depth will cause less latency.

Definition 3.15. *Let L be a matrix in* $M_k(M_n(\mathbb{F}_\mathbf{2}))$*. Then L is called an Maximum Distance Separable (MDS) matrix if all square sub-matrices* $G \in M_t(M_n(\mathbb{F}_\mathbf{2}))$ *of L are of full rank for* $1 \leq t \leq k$*.*

An MDS matrix is a matrix representing a function with specific diffusion properties that have various applications in cryptography. If an $m \times m$ square matrix M is an MDS matrix, then the branch number of M equals $m + 1$, which is the maximum branch number that an $m \times m$ matrix can have.

3.4 Related Works

The generalized structure of the involutory MDS matrix M_{KLSW} studied in [34] costs 84 XOR gates. Then, this method is used to construct an involutory MDS matrix G with lower XOR gates in [36]. They apply the SLP heuristic to G and see that G can be implemented with 80 XOR gates.

The binary representation of the involutory MDS matrix proposed by [34] in $M_4(M_2(\mathbb{F}_{2^4}))$ is;

$$M_{KLSW} = \begin{bmatrix} I_4 & 0 & C & 0 & C^2 & 0 & I_4 & 0 \\ 0 & I_4 & 0 & C & 0 & C^2 & 0 & I_4 \\ C & 0 & I_4 & 0 & I_4 & 0 & C^2 & 0 \\ 0 & C & 0 & I_4 & 0 & I_4 & 0 & C^2 \\ C^3 & 0 & C & 0 & I_4 & 0 & C & 0 \\ 0 & C^3 & 0 & C & 0 & I_4 & 0 & C \\ C & 0 & C^3 & 0 & C & 0 & I_4 & 0 \\ 0 & C & 0 & C^3 & 0 & C & 0 & I_4 \end{bmatrix}, \quad \text{where} \quad C = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The involutory MDS matrix M_{KLSW} can be regarded as a matrix in $M_4(GL(8, \mathbb{F}_2))$ as follows:

$$\begin{bmatrix} I_8 & A & A^2 & I_8 \\ A & I_8 & I_8 & A^2 \\ A^3 & A & I_8 & A \\ A & A^3 & A & I_8 \end{bmatrix}$$

The authors in [36] generalize this matrix representation and consider the the following form:

$$G = \begin{bmatrix} I_8 & A^l & A^i & I_8 \\ A^l & I_8 & I_8 & A^i \\ A^j & A^k & I_8 & A^l \\ A^k & A^j & A^l & I_8 \end{bmatrix} = \begin{bmatrix} I_8 & A^l & A^i & I_8 \\ A^l & I_8 & I_8 & A^i \\ A^{i+k} & A^k & I_8 & A^l \\ A^k & A^{i+k} & A^l & I_8 \end{bmatrix}, \quad \text{where } j = i + k.$$

The involutory condition $G^2 = I$ implies $A^{2l} + A^{i+j} + A^k = 0_8$ and $A^{i+k} + A^j = 0_8$. Under this condition, the authors in [36] obtain further generalizations on the structure of G among 4×4 matrices whose entries are from $GL(8, \mathbb{F}_2)$. While generalizing the matrix, smallness of the DXC is also taken into account. DXC(G) = $w(G)$ - 32 = $4w(A^l) + 2w(A^i) + 2w(A^k) + 2w(A^{i+k}) + 6w(I_8)$ - 32. The article [36] heuristically states $w(A^t)$ increases along with $|t|$ and in order to have small DXC the authors prefer $|i|, |j|, |k|$ and $|l|$ to be small.

A has been chosen as;

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad \text{which is the companion matrix of } x^8 + x^2 + 1.$$

The minimal polynomial of A is $x^8 + x^2 + 1$. Thus, A need to satisfy this equality. From the common solution of condition, $A^{2l} + A^{i+j} + A^k = 0_8$, comes from involutory property and condition, $A^8 + A^2 + I_8 = 0_8$, comes from minimal polynomial solutions of i, j, k, l can be identified. The authors multiply both sides of the equation $A^8 + A^2 + I_8 = 0_8$ with A^d to solve the set equality, $\{A^{8+d}, A^{2+d}, A^d\} = \{A^{2l}, A^{2i+k}, A^k\}$, for d.

Since the aim is to generate low-cost MDS matrices, while searching for the matrix A, smallness of the DXC of the matrix G is also taken into account. DXC(G) = $w(G)$ - 32 = 4$w(A^l)$ + 2$w(A^i)$ + 2$w(A^k)$ + 2$w(A^{i+k})$ + 6$w(I_8)$ - 32. As the authors of the article intuitively argue that $w(A^t)$ increases along with $|t|$, authors considered DXC while searching for exponents of A. In order to have small DXC such i, j, k, l values are preferred that the sum $4|l| + 2|i| + 2|k| + 2|i + j|$ is minimized. This condition on i, j, k and l is derived from the equation of DXC.

The article has identified some lighter involutory MDS matrices, which can be implemented with only 78 XOR gates with depth 4, whereas the previously best-known one requires 84 XOR gates.

In [36], it was also proven that the depth of a 32×32 MDS matrix with branch number 5 is at least 3. Then they modified Boyar's SLP-heuristic algorithm according to the depth awareness to limit the depth of output circuits. They also give the formula to compute the minimum achievable depth of a circuit. They present some examples of lightweight involutory MDS matrices. One of the best matrices they identify has 88 XOR gates with circuit depth 3.

In Algorithm 1, a modified version of Boyar's algorithm with circuit depth awareness is given.

Algorithm 1 Boyar's SLP with circuit depth awareness [36]

Input: The m $\times$ n binary matrix M which xor gate number and dept desired to be calculated

Output: S = $[x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2}, \ldots, x_{n+l}]$ such that $d(x_j) \leq H$ for all j, and for any y_k with $1 \leq k \leq m$, y_k can be computed by one element in S_l, where $x_{n+j} = x_a + x_b$, $x_a, x_b \in \{x_1, \ldots, x_{n+j-1}\}$ for $j \geq 1$.

/*Initialization*/

$S \leftarrow [x_1, x_2, \ldots, x_n]$ /* The input */

$D \leftarrow [0, 0, \ldots, 0]$ /* D[i] represents the circuit depth of S[i] */

$\Delta \leftarrow [\delta_H(S, y_1), \ldots, \delta_H(S, y_m)]$ /* Represents the distances */

if $\Delta[i] = \infty$ for some i **then**

Infeasible /* It is not possible to implement M within the bounds of depth, H */

$j \leftarrow n$

while $\Delta \neq 0$ **do**

$j \leftarrow j + 1$

if $\exists (x'_a, x'_b) \in S$ such that $y_t = x'_a 4 x'_b$ for some $t \in \{1, \ldots, m\}$ **then**

$(x_a, x_b) \leftarrow (x'_a, x'_b)$

else if then

$(x_a, x_b) \leftarrow$ Pick (S, D, H)

$x_j \leftarrow x_a + x_b$

$S \leftarrow S \cup [x_j]$

depth$(x_j) \leftarrow$ max(D[a], D[b]) + 1 /* Depth computation of x_j */

D $\leftarrow$ D $\cup$ [depth(x_j)]

$\Delta \leftarrow [\delta_H(S, y_1), \ldots, \delta_H(S, y_m)]$ /* Update the distances */

return S

The complete list of matrices found in [36] is given below in Table 3.1.

Table 3.1: The number of matrices having weight between 148-172 accoring to DXC, XOR gates and Depth given in [36].

weight	number of matrices	DXC	max. XOR Gate	min. XOR Gate	min. Depth
148	18	116	80	80	4
149	48	117	80	80	4
150	72	118	83	80	4
151	48	119	84	83	4
152	60	120	87	83	4
153	72	121	84	80	4
154	84	122	86	80	4
155	24	123	87	86	5
156	72	124	87	86	4
157	96	125	94	82	5
158	156	126	90	80	4
159	0	-	-	-	-
160	210	128	90	**78**	**4**
161	144	129	84	79	4
162	204	130	89	79	4
163	192	131	91	79	5
164	300	132	93	78	4
165	312	133	88	79	5
166	324	134	93	80	4
167	336	135	94	80	5
168	600	136	99	78	4
169	384	137	97	79	4
170	504	138	98	80	4
171	528	139	99	81	4
172	762	140	102	79	4

One of the optimal matrices reached by Li et. al. [36] is given below as an example.

$$H = \begin{bmatrix} I_8 & I_8 & I_8 & A^4 \\ A^4 & I_8 & A^6 & A^2 \\ A^2 & A^4 & I_8 & A^2 \\ A^6 & I_8 & A^2 & I_8 \end{bmatrix}, \quad A \text{ is the companion matrix of } x^8 + x^2 + 1.$$

3.5 Our Contribution

In order to construct MDS matrices, several methods have been applied. There are search-based, direct generation, and generation methods using custom matrices. In this article, sub-field construction methods have been used to obtain involutory MDS matrices.

In addition to the features that make MDS matrices preferable, there are other factors that can be considered while applying these methods. Low implementation cost is one of them. MDS matrices are generally matrices with high implementation costs depending on the diffusion properties they have. The sub-field construction method is used with the motivation that the Matrix has a low implementation cost. Thus, while obtaining good diffusion properties, the efficiency of an algorithm can be increased.

Moreover, the fact that the matrix is involutory makes it more appealing. Thanks to the involutory nature of the matrix, the processing time and implementation cost of the matrix are equal in encryption and decryption processes. This feature not only benefits us in cryptanalysis, especially in timing attacks, However, it also provides low implementation costs by using the same operations used during encryption in decryption. The involutory feature also ensures that storing one matrix is sufficient and thus takes up little memory space.

The following form of an involutory matrix, which studied in [36], is considered in this thesis, see Section 3.4;

$$G = \begin{bmatrix} I_8 & A^l & A^i & I_8 \\ A^l & I_8 & I_8 & A^i \\ A^j & A^k & I_8 & A^l \\ A^k & A^j & A^l & I_8 \end{bmatrix},$$

where A has chosen to be the companion matrix of $x^8 + x^2 + 1$.

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The matrix G is an element of $M_4(GL(8, \mathbb{F}_2))$, and the involutory condition implies following restrictions:

$$A^{2l} + A^{i+j} + A^k = 0_8 \quad \text{and} \quad A^{i+k} + A^j = 0_8.$$

If we arrange the matrix according to the condition;

$$A^{i+k} + A^j = 0_8 \;\; \rightarrow \;\; A^{i+k} = A^j \;\; \rightarrow \;\; i + k = j,$$

the following form would be obtained:

$$G = \begin{bmatrix} I_8 & A^l & A^i & I_8 \\ A^l & I_8 & I_8 & A^i \\ A^{i+k} & A^k & I_8 & A^l \\ A^k & A^{i+k} & A^l & I_8 \end{bmatrix}.$$

Once this form is obtained, matrices with good implementation properties are tried to find by searching through i, j, k, l. In order to have good implementation properties direct XOR count (DXC) need to be checked.

DXC(G) = $w(G) - 32 = 4w(A^l) + 2w(A^i) + 2w(A^k) + 2w(A^{i+k}) + 48 - 32$.

Remark. *Heuristically $w(A^t)$ seems to increase along with $|t|$. Although, this is valid up to some t, $|t|$ does not effect the $w(A^t)$ as desired due to the non-linearity and irregularity of the increase. Thus, there is no need for limitations for $|i|$, $|j|$, $|k|$, $|l|$.*

According to Definition 3.4 the minimal polynomial of A is also $x^8 + x^2 + 1$. Hence, as A need to satisfy this equation and the conditions from the involutory property, which are $A^{2l} + A^{2i+k} + A^k = 0_8$, ($A^{2l} + A^{i+j} + A^k = 0_8$, as $2i + k = i + j$), and $A^{i+k} + A^j = 0_8$.

In order to find all i, j, k, l values, binomials and trinomials dividing $x^8 + x^2 + 1$ have been searched. The research has done in Laurent Polynomial Ring in order not to miss any negative i, j, k, l. The found binomials are set equal to $A^{i-j+k} + I$, which is obtained by putting in parentheses of A^j to see it in binomial form. Likewise, the found trinomials are set equal to $A^{2l-k}+A^{i+j-k}+I$, obtained by putting in parentheses A^k. Similarly, $A^{2l-i-j} + A^{k-i-j} + I$, is obtained by $A^{i+j}(A^{2l-i-j} + I + A^{k-i-j})$ and $A^{i+j-2l} + A^{k-2l} + I$, is obtained by $A^{2l}(I + A^{i+j-2l} + A^{k-2l})$ need to be checked. The related pseudo-code about the search of i, j, k, l values is given in Algorithm 2 and the pseudo-code for matrix generation from found i, j, k and l values is given in Algorithm 3. SageMath [3] has been used to run the algorithms.

Fifty-one matrices whose weights are between 144 and 172 have been found. Table 3.2 shows the number of matrices according to their weights, the DXC and the maximum and minimum XOR Gate results from the Boyar's SLP algorithm [19].

Algorithm 2 Division on Laurent Polynomial Ring [1]

Input: p = $x^8 + x^2 + 1$ /* minimal polynomial of A */

Output: i, j, k, l values s.t binomials and trinomials obtained form involutory condition divides the minimal polynomial of A /* exponents of A */

/*Initialization*/

$R[x]$ = Laurent Polynomial Ring(GF(2))

$S_1 \leftarrow (\ )$

$S_2 \leftarrow [-240, -210, -180, \ldots, 240]$

/*Division*/

for $ii \in [-2^8, 2^8]$ **do**

  for $jj \in [-2^8, 2^8]$ **do**

    $t = x^{ii} + x^{jj} + 1$

    set (q, r) s.t. $p = t \cdot q + r$ on $R[x]$

    if $r = 0$ **then**

      $s = (ii, jj)$

      $S_1 = S_1 \cup$ Set($[s]$)

for $iii \in S_2$ **do**

  equation$_1 \leftarrow i + k - j = iii$ mod 30

  for $(x, y) \in S_1$ **do**

    equation$_2 \leftarrow 2l - k = x$ mod 30

    equation$_3 \leftarrow i + j - k = y$ mod 30

    Solve equation$_1$, equation$_2$ and equation$_3$ for i, j, l

return i, j, l

Algorithm 3 Matrix Generation [2]

Input: The solution set S contains i, j, k, l values s.t binomials and trinomials obtained form involutory condition divides the minimal polynomial of A, obtained from previous algorithm 2

Output: MDS matrices and their weight, DXC and distance

/*Initialization*/

$M = M_4(GL(8, \mathbb{F}_2))$

$A \in M \leftarrow$ characteristic polynomial of $x^8 + x^2 + 1$

$I \leftarrow I_8 \in M$

/*Generation*/

for $(i, j, l) \in S$ **do**

k = j - i

$$G = \begin{bmatrix} I_8 & A^l & A^i & I_8 \\ A^l & I_8 & I_8 & A^i \\ A^{i+k} & A^k & I_8 & A^l \\ A^k & A^{i+k} & A^l & I_8 \end{bmatrix}$$

$D = [\ ]$

for $i \in [0, \ldots, 30]$ **do**

for $j \in [0, \ldots, 30]$ **do**

if $G_{i,j} = 1$ **then**

weight = weight + 1 /* DXC = Weight - 32 */

d = d + 1

D.append(d) /* distance = D */

d = 0

return Matrices, Weight, DXC and Distance

Table 3.2: Matrices found with their weights, DXC and minimum & maximum XOR gate numbers.

weight	number of matrices	DXC	max. XOR Gate	min. XOR Gate
144	4	112	84	83
146	0	-	-	-
148	6	116	90	81
150	1	118	92	92
152	5	120	98	85
154	1	122	95	95
156	3	124	86	78
158	3	126	97	80
160	4	128	96	**64**
162	1	130	92	92
164	3	132	94	81
166	2	134	92	92
168	14	136	102	72
170	0	-	-	-
172	4	140	103	70

As seen from the table, the best SLP result belongs to matrices with a weight of 160. Two matrices have the minor SLP result, which is 64, and they are given below.

$$M_{(i:-2,\ j:-2,\ k:0,\ l:6)} = \begin{bmatrix} I_8 & A^6 & A^{-2} & I_8 \\ A^6 & I_8 & I_8 & A^{-2} \\ A^{-2} & I_8 & I_8 & A^6 \\ I_8 & A^{-2} & A^6 & I_8 \end{bmatrix}$$

$$M_{(i:6,\ j:6,\ k:0,\ l:-2)} = \begin{bmatrix} I_8 & A^{-2} & A^6 & I_8 \\ A^{-2} & I_8 & I_8 & A^6 \\ A^6 & I_8 & I_8 & A^{-2} \\ I_8 & A^6 & A^{-2} & I_8 \end{bmatrix}$$

It can also be deduced from the table that the weight only gives an upper bound for the number of XOR Gates. Less weight does not mean less XOR gate number. Intuitively, the only factor that affects XOR gate number directly in the distribution

of the 1's and 0's in the matrix. The distribution of 1's and 0's may be checked using the distance vector.

To understand if a limitation can be made, the distance over 8×8 matrices have been examined. We tried to see a classification about what matrices have better results, but no results have been seen. This led us to the idea that good and bad SLP results could also be obtained from a randomly selected from the matrix group.

Authors in [36] have proven that *an MDS matrix* $A \in M_4(GL(8, \mathbb{F}_2))$ *with branch number 5 has at least circuit depth 3.* Thus, the minor circuit depth is satisfied for all matrices given in the table.

In this thesis, so far, the lightest 32×32 involutory MDS matrices, whose depth is 3, with branch number 5, have been found. When finding these matrices, the only involutory constraint was used. In the calculation of XOR gates and depths, a modified version of Boyar's SLP algorithm from the article [36] was used. The results from the modified algorithm of the two lightest matrices were given in Appendix A, Table A.1 and Table A.2.

3.6 Concluding Remarks

Both matrices found in this thesis and AES's Mix Column matrix are 32×32 matrices when represented as binary. When AES's Mix-Column matrix is seen as a 4×4 MDS matrix, it takes its entries from the field $\mathbb{F}_{2^8} = \mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$, and performs the multiplication in 8-bit words. On the other hand, the matrices found in this thesis are in 4×4 form, they take the inputs among 8×8 invertible binary matrices and perform the multiplication among them. There is no difference between our matrices' input and word sizes and AES's, just the operations done in different algebraic structures. The advantage of the multiplication that AES uses in its Mix-Column layer is that multiplying by 02, i.e. x, is equal to the combination of a shift and addition operation. Multiplication by x in the studied field is equal to multiplying by the companion matrix of the irreducible polynomial used to generate the field. Using a method called x-time, the cost of the matrix multiplication can be expressed as look-ups and remaining XOR's. For instance, in the AES, since the irreducible

polynomial is $x^8 + x^4 + x^3 + x + 1$, the x-time operation can be implemented at the byte level as a left shift and a subsequent conditional bitwise XOR with 1B. As a result, the 4×4 byte MDS matrix in the AES consumes two x-time, and 4 XOR's per row of the matrix, or eight x-time and 16 XOR's per MixColumns matrix, or 32 x-time and 64 XOR's per round [12]. Furthermore, as stated in the article [9], it is possible to implement MixColumns transformation for one round with 16 x-time and 60 XOR's.

As a result, the matrices found in this study are superior in XOR gate number and depth compared to AES's without using x-time. However, it is not easy to implement matrices generated in this thesis using x-time since matrix A is a companion matrix of a reducible polynomial. Nevertheless, since the matrices generated in this thesis are involutory, they are still superior to the matrix of AES in this respect. If matrix A is a companion matrix of an irreducible polynomial, multiplication by A would be equivalent to multiplication by x in the field generated using the same irreducible polynomial. We choose the polynomial as a reducible one to have a matrix with a lower weight, thus a lower direct XOR gate number. In fact, when examining the resultant matrices, there is no definite relationship between direct XOR count and XOR gate number; more direct XOR count does not mean more XOR gate numbers. Hence, if the matrix A is chosen as a companion matrix of an irreducible polynomial, its multiplication can be expressed in terms of x-time and compared with AES's MDS matrix. In conclusion, choosing the polynomial of a matrix as an irreducible polynomial would affect the studied algebraic structure and might provide better implementation results.

CHAPTER 4

CONCLUSION

The increasing usage of embedded devices led to much research on adapting existing cryptographic primitives for the low power and energy constraints associated. With the increasing usage of resource-limited devices, lightweight block ciphers aim at providing energy-efficient ways to ensure confidentiality for fixed-size block messages. Block ciphers are under symmetric cryptographic algorithms and can be created in several structures, mainly Feistel networks and Substitution Permutation Networks (SPN). In these structures, linear transformations that serve as permutations to create diffusion can be used. Maximum distance separable (MDS) matrices are the linear transformations that provide great diffusion.

In this thesis, MDS matrices, allowing a very efficient implementation, has been generated. Among the matrices obtained, the ones that give the best implementation results are given in the Appendix A, along with the implementation results specifying the xor gate number and depth. The **depth** of the 32×32 matrix that gives the best results is **3**, while the **number of xor gates is 64**. In the literature review, no lighter matrix was found in the same dimensions. The matrix is not only lightweight but also provides the involutory condition. This feature reduces the number of matrices that need to be kept in memory, as well as the number of new operations to be made, as it allows the same matrix, hence the same operations, to be used in both encryption and decryption processes.

In this thesis, the MDS matrix generation method in the article [36] is studied, and the search area is increased, as mentioned in Section 3.5, by taking it one step further. By expanding the search space, matrices with better XOR gate number and

depth than the matrices found in the inspired article were found. The matrices found were then examined by running Boyar's algorithm [19] and classifying them according to XOR gate number and depth. Matrices were examined in terms of hamming weights, weight distributions and distances. In the examinations, no common feature was observed in matrices with good XOR gate number and dept. According to the observations, the only thing that can be stated is that the number of direct XOR counts creates an upper bound to the number of XOR gates.

These matrices can widely be used in future cryptographic designs to reduce the implementation cost without compromising security. These matrices will allow the permutation layers of lightweight algorithms to provide solid diffusion. Therefore, the usage of these matrices would make lightweight algorithms more resistant. Furthermore, when we reconsider the matrices obtained with the in formations from Chapter 2, it could be seen that it is possible to implement these matrices in resource-limited systems.

Matrices found in this thesis can be used to provide strong diffusion in any cryptographic algorithm with a block length of 32 and multiples. To perform this matrix multiplication, the 32-bit input is included in the process in the form of 4 8-bit words. If desired, this multiplication could also be performed in binary form as with whole 32-bit input. Considering that MDS matrices are not only used in block ciphers, the matrices found could be used wherever MDS matrices are used. In addition, since the matrices found are involutory, using the same matrix, that is, the same operations in both decryption and encryption, will provide a significant advantage in terms of both memory and speed.

REFERENCES

[1] Division On Laurent Polynomial Ring, https://github.com/gokceyetiser/Division-onLaurent-Polynomial-Ring/blob/main/Div_Laurent.Sage.

[2] Generating Matrices, https://github.com/gokceyetiser/Division-onLaurent-Polynomial-Ring/blob/main/Generate_Matrices.Sage.

[3] SageMath, https://www.sagemath.org/.

[4] Weatherley R Lightweight Cryptography Primitives, GitHub repository, https://github.com/rweather/lightweight-crypto, 2021, accessed: 2021-08-30.

[5] M. D. Aagaard and N. Zidaric, ASIC benchmarking of round 2 candidates in the NIST lightweight cryptography standardization process, IACR Cryptol. ePrint Arch, p. 49, 2021.

[6] M. R. Albrecht, B. Driessen, E. B. Kavun, G. Leander, C. Paar, and T. Yalçın, Block ciphers–focus on the linear layer (feat. pride), in *Annual Cryptology Conference*, pp. 57–76, Springer, 2014.

[7] G. Alvarez, D. De la Guia, F. Montoya, and A. Peinado, Akelarre: a new block cipher algorithm, 1996.

[8] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, Camellia: A 128-bit block cipher suitable for multiple platforms—design and analysis, in *International workshop on selected areas in cryptography*, pp. 39–56, Springer, 2000.

[9] F. Asian, M. Sakalli, B. Asian, and S. Bulut, A new involutory 4 x 4 MDS matrix for the AES-like block ciphers, International Review on Computers and Software, 6(1), pp. 96–103, 2011.

[10] R. Avanzi, *A Salad of Block Ciphers The State of the Art in Block Ciphers and their Analysis*, Attribution-NonCommercial-NoDerivatives, 2017.

[11] S. Banik, A. Bogdanov, T. Isobe, K. Shibutani, H. Hiwatari, T. Akishita, and F. Regazzoni, Midori: A block cipher for low energy, in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 411–436, Springer, 2015.

[12] P. Barreto and V. Rijmen, The Khazad legacy-level block cipher, Primitive submitted to NESSIE, 97(106), 2000.

[13] P. Barreto, V. Rijmen, et al., The Whirlpool hashing function, in *First open NESSIE Workshop, Leuven, Belgium*, volume 13, p. 14, Citeseer, 2000.

[14] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, The SIMON and SPECK lightweight block ciphers, in *Proceedings of the 52nd Annual Design Automation Conference*, pp. 1–6, 2015.

[15] C. Beierle, *Design and analysis of lightweight block ciphers: a focus on the linear layer*, Ph.D. thesis, Ruhr University Bochum, Germany, 2018.

[16] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe, PRESENT: An ultra-lightweight block cipher, in *International workshop on cryptographic hardware and embedded systems*, pp. 450–466, Springer, 2007.

[17] J. Borghoff, A. Canteaut, T. Güneysu, E. B. Kavun, M. Knezevic, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, et al., PRINCE-a low-latency block cipher for pervasive computing applications, in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 208–225, Springer, 2012.

[18] J. Boyar, P. Matthews, and R. Peralta, On the shortest linear straight-line program for computing linear forms, in E. Ochmański and J. Tyszkiewicz, editors, *Mathematical Foundations of Computer Science 2008*, pp. 168–179, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, ISBN 978-3-540-85238-4.

[19] J. Boyar, P. Matthews, and R. Peralta, Logic minimization techniques with applications to cryptology, Journal of Cryptology, 26(2), pp. 280–312, 2013.

[20] G. Crump, What is Latency? And How is it Different from IOPS?, `https://storageswiss.com/2013/12/10/what-is-latency-and-how-is-it-different-from-iops/`, 2021, accessed: 2021-08-30.

[21] J. Daemen, L. Knudsen, and V. Rijmen, The block cipher square, in *International Workshop on Fast Software Encryption*, pp. 149–165, Springer, 1997.

[22] J. Daemen and V. Rijmen, *The design of Rijndael*, volume 2, Springer, 2002.

[23] C. De Canniere, O. Dunkelman, and M. Knežević, KATAN and KTANTAN-a family of small and efficient hardware-oriented block ciphers, in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 272–288, Springer, 2009.

[24] C. De Canniere and B. Preneel, Trivium, in *New stream cipher designs*, pp. 244–266, Springer, 2008.

[25] S. Duval and G. Leurent, MDS matrices with lightweight circuits, IACR Transactions on Symmetric Cryptology, 2018(2), pp. 48–78, 2018.

[26] D. L. G. Filho, P. S. Barreto, and V. Rijmen, The MAELSTROM-0 hash function, 2006.

[27] P. Gauravaram, L. R. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schläffer, and S. S. Thomsen, Grøstl-a SHA-3 candidate, in *Dagstuhl Seminar Proceedings*, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2009.

[28] J. Guo, T. Peyrin, and A. Poschmann, The PHOTON family of lightweight hash functions, in *Annual Cryptology Conference*, pp. 222–239, Springer, 2011.

[29] J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw, The LED block cipher, in *International workshop on cryptographic hardware and embedded systems*, pp. 326–341, Springer, 2011.

[30] M. Hell, T. Johansson, and W. Meier, Grain: a stream cipher for constrained environments, International journal of wireless and mobile computing, 2(1), pp. 86–93, 2007.

[31] D. Hong, J.-K. Lee, D.-C. Kim, D. Kwon, K. H. Ryu, and D.-G. Lee, LEA: A 128-bit block cipher for fast encryption on common processors, in *International Workshop on Information Security Applications*, pp. 3–27, Springer, 2013.

[32] P. Junod and S. Vaudenay, Perfect diffusion primitives for block ciphers, in *International Workshop on Selected Areas in Cryptography*, pp. 84–99, Springer, 2004.

[33] M. Khairallah, T. Peyrin, and A. Chattopadhyay, Preliminary hardware benchmarking of a group of round 2 NIST lightweight AEAD candidates., IACR Cryptol. ePrint Arch., 2020, p. 1459, 2020.

[34] T. Kranz, G. Leander, K. Stoffelen, and F. Wiemer, Shorter linear straight-line programs for MDS matrices, IACR Transactions on Symmetric Cryptology, pp. 188–211, 2017.

[35] J. Lacan and J. Fimes, Systematic MDS erasure codes based on Vandermonde matrices, IEEE Communications Letters, 8(9), pp. 570–572, 2004.

[36] S. Li, S. Sun, C. Li, Z. Wei, and L. Hu, Constructing low-latency involutory MDS matrices with lightweight circuits, IACR Transactions on Symmetric Cryptology, pp. 84–117, 2019.

[37] K. Mohajerani, R. Haeussler, R. Nagpal, F. Farahmand, A. Abdulgadir, J.-P. Kaps, and K. Gaj, FPGA benchmarking of round 2 candidates in the NIST lightweight cryptography standardization process: Methodology, metrics, tools, and results., IACR Cryptol. ePrint Arch., 2020, p. 1207, 2020.

[38] J. Nakahara Jr and E. Abrahao, A new involutory MDS matrix for the AES, Int. J. Netw. Secur., 9(2), pp. 109–116, 2009.

[39] M. J. Renner S, Pozzobon E, LWC Benchmark, GitHub repository, `https://lab.las3.de/gitlab/lwc/compare`, 2021, accessed: 2021-08-30.

[40] V. Rijmen, J. Daemen, B. Preneel, A. Bosselaers, and E. De Win, The cipher SHARK, in *International Workshop on Fast Software Encryption*, pp. 99–111, Springer, 1996.

[41] M. T. Sakalli, S. Akleylek, K. Akkanat, and V. Rijmen, On the automorphisms and isomorphisms of MDS matrices and their efficient implementations, Turkish Journal of Electrical Engineering & Computer Sciences, 28(1), pp. 275–287, 2020.

[42] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson, Two sh: a 128-bit block cipher, AES submission, 1998.

[43] B. Schneier, J. Kelseyy, D. Whitingz, D. Wagnerx, C. Hall, and N. Ferguson, New results on the two sh encryption algorithm, 1999.

[44] C. P. Schnorr and S. Vaudenay, Black box cryptanalysis of hash networks based on multi-permutations, in *Workshop on the Theory and Application of of Cryptographic Techniques*, pp. 47–57, Springer, 1994.

[45] C. E. Shannon, Communication theory of secrecy systems, The Bell system technical journal, 28(4), pp. 656–715, 1949.

[46] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai, Piccolo: an ultra-lightweight blockcipher, in *International workshop on cryptographic hardware and embedded systems*, pp. 342–357, Springer, 2011.

[47] T. Shirai, K. Shibutani, T. Akishita, S. Moriai, and T. Iwata, The 128-bit blockcipher CLEFIA, in *International workshop on fast software encryption*, pp. 181–195, Springer, 2007.

[48] M. S. Turan, K. McKay, D. Chang, C. a Calık, L. Bassham, J. Kang, and J. Kelsey, Status report on the second round of the NIST lightweight cryptography standardization process, 2021.

[49] S. Vaudenay, On the need for multipermutations: Cryptanalysis of MD4 and SAFER, in *International Workshop on Fast Software Encryption*, pp. 286–297, Springer, 1994.

[50] D. Watanabe, S. Furuya, H. Yoshida, K. Takaragi, and B. Preneel, A new keystream generator MUGI, IEICE transactions on fundamentals of electronics, communications and computer sciences, 87(1), pp. 37–45, 2004.

[51] D. J. Wheeler and R. M. Needham, TEA, a tiny encryption algorithm, in *International workshop on fast software encryption*, pp. 363–366, Springer, 1994.

[52] A. Youssef, S. E. Tavares, and H. Heys, A new class of substitution-permutation networks, in *Workshop on Selected Areas in Cryptography, SAC*, volume 96, pp. 132–147, 1996.

[53] L. Zhou, L. Wang, and Y. Sun, On efficient constructions of lightweight MDS matrices, IACR Transactions on Symmetric Cryptology, pp. 180–200, 2018.

APPENDIX A

LIGHTEST MATRICES OBTAINED IN CHAPTER 3

Table A.1: The matrix $M_{(i:-2,\ j:-2,\ k:0,\ l:6)}$, with weight 160, DXC 128, SLP 64, depth 3.

No	Operation	Depth	No	Operation	Depth	No	Operation	Depth
1	$t_1 = x_0 + x_{24}$	1	23	$t_{23} = t_2 + t_3$	2	45	$t_{45} = x_{10} + t_{44}\ [y_2]$	3
2	$t_2 = x_{10} + x_{18}$	1	24	$t_{24} = x_2 + t_{23}\ [y_{10}]$	3	46	$t_{46} = x_{18} + t_{44}\ [y_{26}]$	3
3	$t_3 = x_4 + x_{28}$	1	25	$t_{25} = x_{26} + t_{23}\ [y_{18}]$	3	47	$t_{47} = t_{10} + t_{11}$	2
4	$t_4 = x_{14} + x_{22}$	1	26	$t_{26} = t_3 + t_4$	2	48	$t_{48} = x_{14} + t_{47}\ [y_6]$	3
5	$t_5 = x_1 + x_{25}$	1	27	$t_{27} = x_{12} + t_{26}\ [y_4]$	3	49	$t_{49} = x_{22} + t_{47}\ [y_{30}]$	3
6	$t_6 = x_{11} + x_{19}$	1	28	$t_{28} = x_{20} + t_{26}\ [y_{28}]$	3	50	$t_{50} = t_{11} + t_{12}$	2
7	$t_7 = x_5 + x_{29}$	1	29	$t_{29} = t_5 + t_6$	2	51	$t_{51} = x_4 + t_{50}\ [y_{12}]$	3
8	$t_8 = x_{15} + x_{23}$	1	30	$t_{30} = x_{17} + t_{29}\ [y_1]$	3	52	$t_{52} = x_{28} + t_{50}\ [y_{20}]$	3
9	$t_9 = x_2 + x_{26}$	1	31	$t_{31} = x_9 + t_{29}\ [y_{25}]$	3	53	$t_{53} = t_{13} + t_{14}$	2
10	$t_{10} = x_8 + x_{16}$	1	32	$t_{32} = t_5 + t_8$	2	54	$t_{54} = x_{25} + t_{53}\ [y_9]$	3
11	$t_{11} = x_6 + x_{30}$	1	33	$t_{33} = x_7 + t_{32}\ [y_{15}]$	3	55	$t_{55} = x_1 + t_{53}\ [y_{17}]$	3
12	$t_{12} = x_{12} + x_{20}$	1	34	$t_{34} = x_{31} + t_{32}\ [y_{23}]$	3	56	$t_{56} = t_{13} + t_{16}$	2
13	$t_{13} = x_3 + x_{27}$	1	35	$t_{35} = t_6 + t_7$	2	57	$t_{57} = x_{11} + t_{56}\ [y_3]$	3
14	$t_{14} = x_9 + x_{17}$	1	36	$t_{36} = x_3 + t_{35}\ [y_{11}]$	3	58	$t_{58} = x_{19} + t_{56}\ [y_{27}]$	3
15	$t_{15} = x_7 + x_{31}$	1	37	$t_{37} = x_{27} + t_{35}\ [y_{19}]$	3	59	$t_{59} = t_{14} + t_{15}$	2
16	$t_{16} = x_{13} + x_{21}$	1	38	$t_{38} = t_7 + t_8$	2	60	$t_{60} = x_{15} + t_{59}\ [y_7]$	3
17	$t_{17} = t_1 + t_2$	2	39	$t_{39} = x_{13} + t_{38}\ [y_5]$	3	61	$t_{61} = x_{23} + t_{59}\ [y_{31}]$	3
18	$t_{18} = x_{16} + t_{17}\ [y_0]$	3	40	$t_{40} = x_{21} + t_{38}\ [y_{29}]$	3	62	$t_{62} = t_{15} + t_{16}$	2
19	$t_{19} = x_8 + t_{17}\ [y_{24}]$	3	41	$t_{41} = t_9 + t_{10}$	2	63	$t_{63} = x_5 + t_{62}\ [y_{13}]$	3
20	$t_{20} = t_1 + t_4$	2	42	$t_{42} = x_{24} + t_{41}\ [y_8]$	3	64	$t_{64} = x_{29} + t_{62}\ [y_{21}]$	3
21	$t_{21} = x_6 + t_{20}\ [y_{14}]$	3	43	$t_{43} = x_0 + t_{41}\ [y_{16}]$	3			
22	$t_{22} = x_{30} + t_{20}\ [y_{22}]$	3	44	$t_{44} = t_9 + t_{12}$	2			

Table A.2: The matrix $M_{(i:6,\ j:6,\ k:0,\ l:-2)}$, with weight 160, DXC 128, SLP 64, depth 3.

No	Operation	Depth	No	Operation	Depth	No	Operation	Depth
1	$t_1 = x_0 + x_{24}$	1	23	$t_{23} = t_2 + t_3$	2	45	$t_{45} = x_{18} + t_{44}\ [y_2]$	3
2	$t_2 = x_{10} + x_{18}$	1	24	$t_{24} = x_{26} + t_{23}\ [y_{10}]$	3	46	$t_{46} = x_{10} + t_{44}\ [y_{26}]$	3
3	$t_3 = x_4 + x_{28}$	1	25	$t_{25} = x_2 + t_{23}\ [y_{18}]$	3	47	$t_{47} = t_{10} + t_{11}$	2
4	$t_4 = x_{14} + x_{22}$	1	26	$t_{26} = t_3 + t_4$	2	48	$t_{48} = x_{22} + t_{47}\ [y_6]$	3
5	$t_5 = x_1 + x_{25}$	1	27	$t_{27} = x_{20} + t_{26}\ [y_4]$	3	49	$t_{49} = x_{14} + t_{47}\ [y_{30}]$	3
6	$t_6 = x_{11} + x_{19}$	1	28	$t_{28} = x_{12} + t_{26}\ [y_{28}]$	3	50	$t_{50} = t_{11} + t_{12}$	2
7	$t_7 = x_5 + x_{29}$	1	29	$t_{29} = t_5 + t_6$	2	51	$t_{51} = x_{28} + t_{50}\ [y_{12}]$	3
8	$t_8 = x_{15} + x_{23}$	1	30	$t_30 = x_9 + t_{29}\ [y_1]$	3	52	$t_{52} = x_4 + t_{50}\ [y_{20}]$	3
9	$t_9 = x_2 + x_{26}$	1	31	$t_{31} = x_{17} + t_{29}\ [y_{25}]$	3	53	$t_{53} = t_{13} + t_{14}$	2
10	$t_{10} = x_8 + x_{16}$	1	32	$t_{32} = t_5 + t_8$	2	54	$t_{54} = x_1 + t_{53}\ [y_9]$	3
11	$t_{11} = x_6 + x_{30}$	1	33	$t_{33} = x_{31} + t_{32}\ [y_{15}]$	3	55	$t_{55} = x_{25} + t_{53}\ [y_{17}]$	3
12	$t_{12} = x_{12} + x_{20}$	1	34	$t_{34} = x_7 + t_{32}\ [y_{23}]$	3	56	$t_{56} = t_{13} + t_{16}$	2
13	$t_{13} = x_3 + x_{27}$	1	35	$t_{35} = t_6 + t_7$	2	57	$t_{57} = x_{19} + t_{56}\ [y_3]$	3
14	$t_{14} = x_9 + x_{17}$	1	36	$t_{36} = x_{27} + t_{35}\ [y_{11}]$	3	58	$t_{58} = x_{11} + t_{56}\ [y_{27}]$	3
15	$t_{15} = x_7 + x_{31}$	1	37	$t_{37} = x_3 + t_{35}\ [y_{19}]$	3	59	$t_{59} = t_{14} + t_{15}$	2
16	$t_{16} = x_{13} + x_{21}$	1	38	$t_{38} = t_7 + t_8$	2	60	$t_{60} = x_{23} + t_{59}\ [y_7]$	3
17	$t_{17} = t_1 + t_2$	2	39	$t_{39} = x_{21} + t_{38}\ [y_5]$	3	61	$t_{61} = x_{15} + t_{59}\ [y_{31}]$	3
18	$t_{18} = x_{16} + t_{17}\ [y_0]$	3	40	$t_{40} = x_{13} + t_{38}\ [y_{29}]$	3	62	$t_{62} = t_{15} + t_{16}$	2
19	$t_{19} = x_{16} + t_{17}\ [y_{24}]$	3	41	$t_{41} = t_9 + t_{10}$	2	63	$t_{63} = x_{29} + t_{62}\ [y_{13}]$	3
20	$t_{20} = t_1 + t_4$	2	42	$t_{42} = x_0 + t_{41}\ [y_8]$	3	64	$t_{64} = x_5 + t_{62}\ [y_{21}]$	3
21	$t_{21} = x_{30} + t_{20}\ [y_{14}]$	3	43	$t_{43} = x_{24} + t_{41}\ [y_{16}]$	3			
22	$t_{22} = x_6 + t_{20}\ [y_{22}]$	3	44	$t_{44} = t_9 + t_{12}$	2			

These results may be potentially convenient for the design of lightweight and low-latency symmetric-key algorithms.