

DESIGN AND IMPLEMENTATION OF LIGHTWEIGHT ENCRYPTION
ALGORITHM USING PRINCE CIPHER

LEE JIAH CHUN


A project report submitted in fulfilment of the
requirements for the award of the degree of
Master of Engineering (Computer and Microelectronic Systems)

School of Electrical Engineering
Faculty of Engineering
Universiti Teknologi Malaysia

FEBRUARY 2021

DECLARATION

I declare that this project report entitled “*Design And Implementation Of Lightweight Encryption Algorithm Using Prince Cipher*” is the result of my own research except as cited in the references. The project report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature : 

Name : LEE JIAH CHUN

Date : 15 FEBRUARY 2021

DEDICATION

This thesis is dedicated to my father, who taught me that the best kind of knowledge to have is that which is learned for its own sake. It is also dedicated to my mother, who taught me that even the largest task can be accomplished if it is done one step at a time.

ACKNOWLEDGEMENT

In preparing this thesis, I was in contact with many people, researchers, academicians, and practitioners. They have contributed towards my understanding and thoughts. In particular, I wish to express my sincere appreciation to my main thesis supervisor, Dr. Shahidatul Sadiah binti Abdul Manan, for encouragement, guidance, critics and friendship. Without her continued support and interest, this thesis would not have been the same as presented here.

My fellow postgraduate student should also be recognised for their support. My sincere appreciation also extends to all my colleagues and others who have provided assistance at various occasions. Their views and tips are useful indeed. Unfortunately, it is not possible to list all of them in this limited space. I am grateful to all my family member.

ABSTRACT

Lightweight cryptography is widely deployed on low-resource devices that has limited computing power, low memory size and power resource. With the rising of pervasive computing, more devices are connected online, and new requirement on encryption model that emphasizes on ultra-fast response time is introduced. Most of the available lightweight cryptographies are round-based designs, they are able to achieve high throughput via pipelining the round functions, however the response time is not ideal. The Prince cipher is the first lightweight block cipher developed to speed up the latency of the algorithm. Compare to other block ciphers, the Prince is able to yield low latency with very competitive area utilization, hence it is a promising choice for low-resource devices that emphasize of response time. In this work, the Prince cipher will be designed and synthesize in different implementation including round-per-cycle, single-cycle and reduced multicycle implementations. The synthesis results had suggested that the single-cycle Prince cipher is achievable with almost 40% reduction in encryption latency. This indicates the possibility of instantaneous encryption as the full operation can be performed within a single clock cycle and no warm-up phase is needed. However, the implementation using loop unrolling also introduced larger gate count and therefore the design will have bigger silicon footprint. With the improvement of chip technology, it is possible to absorb the increment in of the gate count in the Prince cipher in exchange for performance. Furthermore, the modern SOC design often involves many-core designs that have high-bandwidth, packet-switched network design. These applications need the data to be processed as fast as possible, hence the conventional high throughput looping approaches are not desirable as they might limit the bandwidth of these high-speed buses within the SOC.

ABSTRAK

Kriptografi ringan digunakan secara meluas pada peranti sumber rendah yang mempunyai kuasa pengkomputeran terhad, saiz memori dan sumber kuasa yang rendah. Dengan peningkatan penggunaan komputer dimana-mana sahaja, lebih banyak peranti mesti disambungkan dalam talian. Oleh itu, masa tindak balas yang sangat pantas bagi sesuatu model penyulitan maklumat sangat diperlukan. Sebilangan besar kriptografi ringan tersedia adalah dengan rekabentuk model berasaskan pusingan, yang mana dapat mencapai kadar hasilan yang tinggi, tetapi mempunyai masa tindak balas yang tidak ideal. Prince sifer adalah model blok sifer pertama yang direka untuk mencapai latensi rendah. Prince sifer dapat menghasilkan latensi rendah dengan penggunaan kawasan yang sangat kompetitif semasa berbanding dengan model sifer yang lain. Ia merupakan calon yang baik untuk peranti sumber rendah yang memerlukan masa tindak balas cepat daripada model blok sifer yang lain. Dalam kajian ini, sintesis Prince sifer akan dijalankan untuk tiga reka bentuk yang berbeza iaitu “round-per-cycle”, “reduced-multicycle” dan “single-cycle”. Hasil sintesis menunjukkan bahawa reka bentuk “single-cycle” boleh dicapai dengan 40% pengurangan latensi. Oleh itu, proses penyulitan secara seketika boleh dicapai dengan reka bentuk ini tanpa keperluan fasa pemanasan. Namun begitu, reka bentuk ini juga memperkenalkan bilangan gate yang banyak dan luas kawasan silikon yang lebih besar. Dengan peningkatan teknologi cip, peningkatan luas kawasan reka bentuk boleh diserap sebagai penukaran prestasi yang lebih tinggi. Selain itu, SOC moden selalunya mengandungi reka bentuk banyak prosesor yang memerlukan rangkaian antara prosesor yang cepat dan keperluan ini boleh dipenuhi dengan reka bentuk “single-cycle”. Aplikasi ini memerlukan data untuk diproses secepat mungkin, oleh itu pendekatan perulangan kadar hasilan tinggi konvensional tidak diutamakan kerana ada kebarangkalian ia menghadkan lebar jalur bas berkelajuan tinggi dalam SOC.

TABLE OF CONTENTS

	TITLE	PAGE
	DECLARATION	iii
	DEDICATION	iv
	ACKNOWLEDGEMENT	v
	ABSTRACT	vi
	ABSTRAK	vii
	TABLE OF CONTENTS	viii
	LIST OF TABLES	x
	LIST OF FIGURES	xi
CHAPTER 1	INTRODUCTION	1
1.1	Introduction	1
1.2	Problem Background	2
1.3	Problem Statement	4
1.4	Research Goal	4
	1.4.1 Research Objectives	4
1.5	Scope of work	5
1.6	Contribution	5
CHAPTER 2	LITERATURE REVIEW	7
2.1	Introduction	7
2.2	Classification of Cipher Implementation	8
	2.2.1 High-Performance System	9
	2.2.2 General Purpose Processor System	9
	2.2.3 Low-Resource Device	9
	2.2.3.1 Software Implementation	10
	2.2.3.2 Hardware Implementation	10
2.3	Lightweight Cryptography	10

2.3.1	Prince cipher	11
2.3.1.1	Round Functions	12
2.4	Previous Work	15
2.5	Limitation	16
2.6	Research Gap	16
CHAPTER 3	RESEARCH METHODOLOGY	19
3.1	Introduction	19
3.1.1	Design Implementations	19
3.1.2	Design Synthesis	20
3.1.3	Performance Comparison	21
3.1.4	Design Optimization	22
3.2	Tools and Platforms	23
CHAPTER 4	RESULTS AND DISCUSSION	24
4.1	Synthesis outcome	24
4.2	Optimization	30
4.2.1	Constraint	31
4.2.2	RTL	32
CHAPTER 5	CONCLUSION AND FUTURE WORKS	36
5.1	Future Works	36
5.2	Conclusion	37
REFERENCES		38

LIST OF TABLES

TABLE NO.	TITLE	PAGE
Table 2.1	4-bit S-box in hexadecimal notation.	12
Table 2.2	RC-dependent constant in hexadecimal notation	14
Table 2.3	Area, critical path latency and throughput comparison for single cycle lightweight ciphers on ASIC (Maene, 2019).	15
Table 4.1	Library cell usage for single cycle implementation at 4ns and 6ns clock period.	27
Table 4.2	Synthesis result comparison for the three implementation of PRINCE cipher.	30
Table 4.3	Synthesis flow comparison between default flow and optimized flow.	31
Table 4.4	Synthesis result of single cycle implementation using different constraint approaches.	32
Table 4.5	Code snippet for 4-bit Substitution box	33
Table 4.6	S-box synthesis result using different coding method.	33
Table 4.7	Result comparison of Round per Cycle implementation with optimized S-box.	34
Table 4.8	Result comparison of Reduced Multicycle implementation with optimized S-box.	34
Table 4.9	Result comparison of Single Cycle implementation with optimized S-box.	35

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE
Figure 2.1	Overview of Cryptography	7
Figure 2.2	Cipher Implementation Taxonomy	8
Figure 2.3	The Prince encryption core	11
Figure 2.4	Shift row operation of linear diffusion layer	12
Figure 2.5	The basic building blocks for the matrix M'	13
Figure 2.6	16x16 involution matrices built from the basic building blocks	13
Figure 3.1	Synthesis tool inputs and output	20
Figure 4.1	Critical path delay for the three implementation using different timing constraints.	25
Figure 4.2	Total cell count for the three implementation using different timing constraints.	26
Figure 4.3	Total power of the three implementations using different timing constraints.	28
Figure 4.4	Energy consumption per bit using different timing constraints.	29

CHAPTER 1

INTRODUCTION

1.1 Introduction

Security plays a special role in modern life. Following the advancement of technology, people are migrating all sort of activities to virtual platform, from shopping, entertainment, socializing, to even business and banking. Technology fills the people's everyday lives, and there is no activity today that does not involve computers and network. In recent years, there is even a growing trend of pervasive computing, where embedded devices are introduced into everyday objects to aid in human life. They are deployed into a huge range of domains, including industries, private and public sectors, important infrastructures, and even portable and wearable applications. These devices are gathering user information using Internet communication to provide various services. These services can include healthcare, smart home systems, smart factory and etc., and they often handle sensitive data such as privacy, secret, safety. Hence, it becomes vital to have security systems in place to protect the users against malicious attack [1].

Generally, the devices that deployed massively in pervasive computing platform come with constraints in terms of computing capability, memory size, as well as power resources. These limitations are causing difficulties in the deployment of complex security systems in the devices. There are many available cryptographies that can provide good security on the sensitive information that handled by the systems however they do not perform well in resource-limited environment. Oftentimes, the security systems cause heavy load on the devices and result in reduced performance. Moreover, there is a persistent need for security systems that entail smaller size and low production cost in these devices. The lightweight cryptography was introduced to tackle the issue faced in common complex security system to provide security in these constrained devices without tampering with the performance of the device.

Lightweight cryptography is a type of encryption method that features smaller silicon footprint and less complex computation. It is suitable for constrained devices that has limitation in computing, memory as well as power resource. The current typical cryptography algorithms usually require high amount of resources in their implementation as they are specifically designed for secure communication in larger systems. This directly translate to bigger silicon footprint and higher implementation cost, which cause these conventional cryptographies not cost-effective for small resource-limited devices. The lightweight cryptography addresses this issue by excluding complex computation and utilize round-based design that is friendly for hardware implementation in terms of area and timing, with some drawback in the security level of the algorithm.

1.2 Problem Background

The lightweight cryptography is targeted towards applications involving constrained devices, including microcontrollers, smart devices, small computers and even sensors. The common characteristics for these constrained devices include limited processing capabilities, low bandwidth and low memory area. Most of the time, these devices are also driven using batteries and they have high requirement on low power consumption to ensure continuity. With the emerging of Internet of Things (IoT) computing environment, a lot of the aforementioned constrained devices are now having the capability to be connected to the Internet. Massive communications are done between each device in seconds. Reliable and secure data transfer are needed, and they need to be carried out as fast as possible.

There are two ways a design is considered fast – a high throughput design, where high amount of operations can be carried out within a given period; or a low latency design, where the waiting time for the result is significantly short. In current context, most of the fast cryptography design are high throughput design where the design is extensively pipelined to achieve high throughput. However, this approach tends to cause the design to have high latency, which can be undesirable. As the rising of pervasive computing, new requirements on the cryptography such as ultra-fast

response times are introduced [2]. The currently high throughput cryptography design is unable to meet this requirement as it might need multiple clock cycles to encrypt a single simple message block. Therefore, a low latency cryptography that can perform instant message encryption is required.

The design of low-latency design poses a challenge in current available cryptosystems. Low latency design is difficult to be realized in software implementation platform as the ciphers could take up to hundreds or more clock cycles to perform the encryption. Stream ciphers are cryptography algorithm that perform encryption on bit-by-bit basis. It is hardly suitable for low-latency design due to high number of clock cycles are needed during the starting phase of the ciphers, making it to be a high throughput and high latency design. This makes block ciphers that operate on a fixed length of block size to be the potential choice for low-latency design.

There are some challenges to be addressed in block ciphers, particularly the they are round-based design where the plaintext has to be fed into the cipher round for multiple times in order to obtain the ciphertext. For example, the AES algorithm requires 10 rounds of encryption operations before the secure ciphertext can be obtained. This means that at least 10 clock cycles are required before the data can be encrypted, making it non-ideal for low-latency design. One approach to tackle the issue is by employing loop-unrolled design to make the whole block ciphers operation to be completed in single cycle or lesser multicycle. However, this might cause a very long critical path in the design and result in slow response time and poor maximum operating frequencies. Moreover, the unrolled design may introduce few times higher gate count which cause higher power consumption and implementation costs. To reduce the gate count for unrolled design, less complex logic is needed in the cipher round. This call for a lightweight block cipher that can provide small area and fast response time.

1.3 Problem Statement

Block cipher is a round based design and the implementations often involve high pipeline stages due to complex logic which hold back the response time. To achieve good latency, lightweight design with single-cycle or reduced multicycle implementation is required.

1.4 Research Goal

Based on the discussed issues, the goal in this research is to design a lightweight block cipher that can perform instantaneous encryption to suit low-resource devices that has requirement for ultra-fast response time. The lightweight block cipher needs to achieve low silicon footprint and low energy consumption as well in single cycle and reduced multicycle implementation.

1.4.1 Research Objectives

The objectives of the research are:

- (a) To design the block cipher in single-cycle and reduced multicycle implementation.
- (b) To analyse the gain in performance matrix in terms of area, latency and throughput for single-cycle and reduced multicycle implementations with regard to normal implementation.
- (c) To evaluate the static power consumption and efficiency of single-cycle and reduced multicycle block ciphers.

1.5 Scope of work

The scope of work includes:

- (a) The Prince is used as the primary algorithm in this research as it is the block cipher that focus on latency reduction.
- (b) Only the encryption portion of the Prince algorithm is implemented and synthesize.
- (c) The implemented design is synthesized using Synopsis Design Vision Version O-2018.06-SP3.
- (d) The design synthesis is based on SAED 32nm Library.

1.6 Contribution

In this research, two extended implementations from the conventional PRINCE cipher - the single cycle and reduced multicycle implementations are designed and synthesized. The designs are able to provide low delay encryption processes through the elimination or reduction or the round function looping. For single cycle implementation, further latency reduction can be achieved. Moreover, since the encryption only requires one clock cycle to perform, almost instantaneous block encryption is possible with the design. Nowadays the design of SOCs often comes with ultra-fast and high bandwidth on-chip network to cope with the multi- and many-cores designs [3]. Hence, the conventional way of having high throughput block cipher via extensive pipelining are not desirable. Instead, the response time of the encryption need to be instantaneous such that all arriving data element can be processed with as little delay as possible. Furthermore, the implementations are realized using loop unrolling, hence it is only natural to introduce large combinational data path in the design. Optimization to further reduce the latency in the design are proposed and proven in this research as well.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Lightweight cryptography is designed with the purpose of extending the application of encryption to the ever-growing computing platforms that employ smart and low-resource devices. A fundamental level of security is a must for these platforms as their applications tend to involve exchange of private or sensitive data. Due to the computational and energy constraints, the installed encryption model should introduce little to none burden in these low-resource devices in order to minimize the impact to the performance and lifetime.

A low-resource device is one of the device categories that is having low computing power, battery supplied, small silicon area and small memory size. The design of cryptography cipher is a balancing act between the achievable security level, implementation cost, execution speed and power consumption. For a lightweight cryptography, certain level of compromise in the security level is acceptable to achieve low cost, fast and energy saving algorithm.

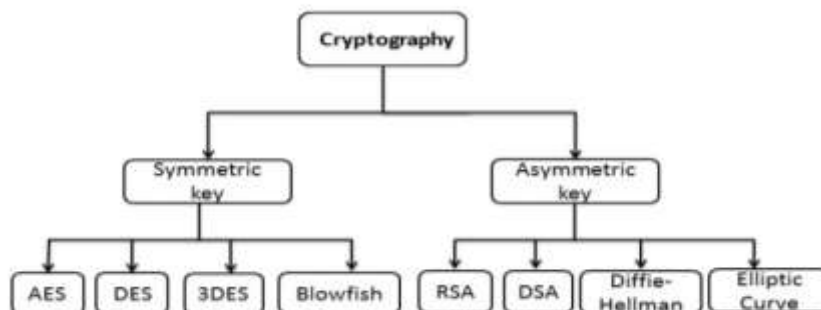


Figure 2.1 Overview of Cryptography

The block ciphers are categorized into two types: symmetric and asymmetric key ciphers. The asymmetric key block ciphers utilize a public key and a hidden private key for encryption and decryption. It offers more security features and safer in comparison to the symmetric key block ciphers that use a shared key among all parties. However, due to extra computational resources required to figure out the hidden private key, the asymmetric key block ciphers are generally more costly and slower, making them to be less preferable for resource-constrained devices. Most of the available security model for hardware implementations are based on symmetric key cryptography.

2.2 Classification of Cipher Implementation

Figure 2.2 illustrates the taxonomy of cipher implementation platform. In overall, the ciphers are implemented in three different platforms and different platforms are having their own requirements that need to be fulfilled. These platforms include high-performance system, general purpose processor system and low-resource device. While the main focus in this research is the low-resource implementation platform, other different platforms are discussed to show the distinctions between each other.

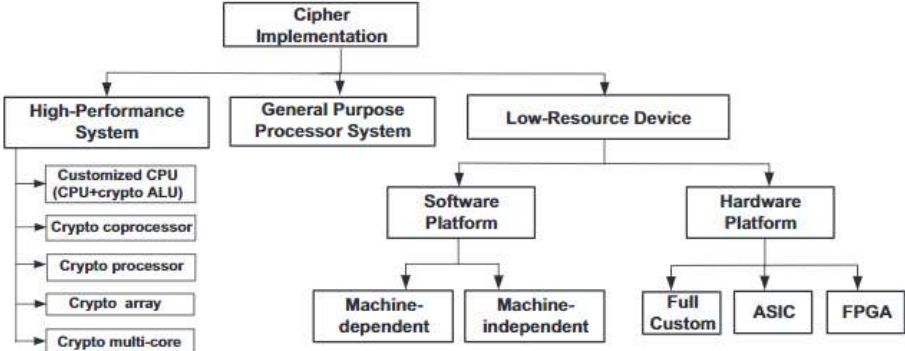


Figure 2.2 Cipher Implementation Taxonomy

2.2.1 High-Performance System

High-performance system must meet three basic requirements: security, throughput and flexibility [3]. This kind of system is often used from military to commercial purpose where powerful cryptography is needed to ensure maximum security. The system often operates in high speed; therefore, the ciphers implementation must be able to cope with the operations of the system. The system may also process complex data that may come in different forms and sizes. Hence flexibility is a special need so that the ciphers are capable to handle the encryption of data in various size. In order to meet these requirement, specialized engines such as customized CPU, crypto processor and coprocessor are usually utilized. Meanwhile, there is lesser concern on area and cost in this platform.

2.2.2 General Purpose Processor System

Unlike high-performance system, no specialized engine is being used for general purpose processor system. High level machine independent code is used to realize the implementation of ciphers. Some popular choices include Java and C. Since the implementation is in fully software, no hardware acceleration technique can be used to boost the performance of the cryptography. The implementations often involve various approaches to push the code execution speed to obtain improvement performance.

2.2.3 Low-Resource Device

There are two types of implementation in low-resource devices, they are software and hardware implementations. In low-resource device, implementation cost is a deciding factor. Most of the deployed design need to fulfill low-cost and energy saving requirement. Typically, the lightweight cryptography is used to implement the security systems for low-resource devices.

2.2.3.1 Software Implementation

Processor and micro-controller are the common platform used for this kind of implementation. Two common ways to implement the coding is using a machine-independent language similar to general purpose processor system, or machine-dependent language like assembly language. Typically, the code will be further optimized to suit the processor. For example, the instruction set architecture of the processor is fully utilized to ensure faster implementation by reducing the execution cycles. Code structure and coding style are another important element that can affect the efficiency of the code and memory footprint of coding in the device.

2.2.3.2 Hardware Implementation

The hardware platform implementation focuses on area reduction, speed and low power similar to software implementation. The common design platforms include ASIC, FPGA and full-custom design. There are many techniques employed in hardware implementation to fulfill the design requirements. Sharing of optimized circuit is one of the methods to reduce the area of the design. Different architectures and logic techniques such as loop unrolling, and pipelining are used to maximize the operating frequency and throughput of the device. Meanwhile, clock gating and power gating is used to minimize the overall energy consumption of the device.

2.3 Lightweight Cryptography

A lightweight block cipher is a cryptography algorithm that is tailored to suit the application of low-resource devices. It needs to fulfill three main requirements, which are: minimum area overhead, low power consumption as well as acceptable level of security [4]. The area overhead need to be small to fulfill the low-cost requirement of low-resource devices. The power consumption needs to be low to

ensure the continuity of the devices that usually has limited power resources. Meanwhile, reasonable level of security is needed to ensure the safety of end-user.

The block ciphers are generally having smaller block size and key size such as 32, 48 or 64 bits and compared to the conventional ciphers that employ 64 or 128 bits. The lightweight block ciphers are also having simplify key schedule and operates in simple encryption operations with more iterations of round. Prince algorithm is one of the block ciphers that meets the specification of lightweight cryptography.

2.3.1 Prince cipher

Prince algorithm is the first cryptography algorithm introduced for the purpose of low latency encryption [5]. It is a 64-bit lightweight block cipher that operates on substitution-permutation network. It has 12 rounds at the core and each round of the Prince cipher consists of sixteen 4-bit parallel s-box, a linear diffusion layer and the addition of the round-dependent constant and a fix key. The encryption operates on 128-bit key. The key is split into two 64-bit keys, where the first key is used during the pre- and post-whitening operations, while the second key is used in the round function. The pre- and post-whitening operation are the addition of key before and after the round functions as shown in Figure 2.3.

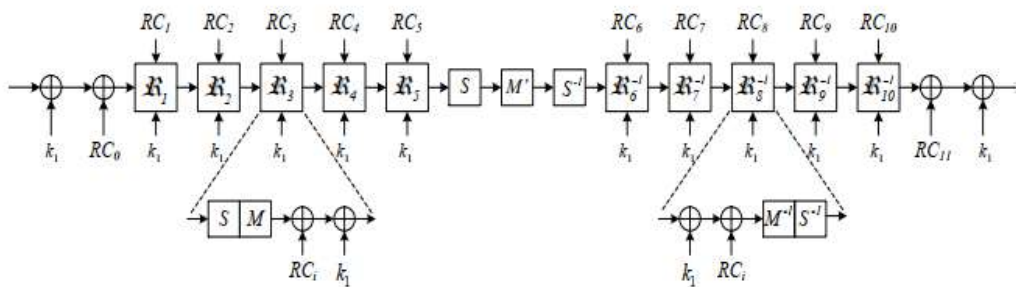


Figure 2.3 The Prince encryption core

2.3.1.1 Round Functions

The round function consists of three parts: The s-box, diffusion layer and key addition. The two main operations in the round function are the substitution and permutation. Prince cipher use 4-bit s-box for the substitution operation. The 4-bit input to the s-box is mapped to another 4-bit value to complete the data substitution.

Table 2.1 4-bit S-box in hexadecimal notation.

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

The linear diffusion layer is mainly used to perform the permutation of data. It consists of two main operation: shift row operation and matrix multiplication with 64×64 matrix M' . The shift row operation in the linear diffusion layer behaves similarly to AES shift row to permutate the 16 nibbles as shown in Figure 2.4.

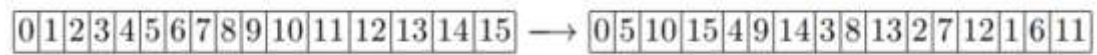


Figure 2.4 Shift row operation of linear diffusion layer

The matrix M' used for the matrix multiplication is a 64×64 involution matrix. The following 4×4 matrices are the basic building blocks for the matrix M' . The number of ones in these basic matrices are kept in minimum to ensure smallest implementation costs.

$$M_0 = \begin{pmatrix} 0000 \\ 0100 \\ 0010 \\ 0001 \end{pmatrix}, M_1 = \begin{pmatrix} 1000 \\ 0000 \\ 0010 \\ 0001 \end{pmatrix}, M_2 = \begin{pmatrix} 1000 \\ 0100 \\ 0000 \\ 0001 \end{pmatrix}, M_3 = \begin{pmatrix} 1000 \\ 0100 \\ 0010 \\ 0000 \end{pmatrix}$$

Figure 2.5 The basic building blocks for the matrix M'

Next, the 4x4 matrices are used to form two 16x16 matrices. These matrices are arranged in such way that they are the row permutation of each other, and they are formed as involution matrices.

$$\hat{M}^{(0)} = \begin{pmatrix} M_0 & M_1 & M_2 & M_3 \\ M_1 & M_2 & M_3 & M_0 \\ M_2 & M_3 & M_0 & M_1 \\ M_3 & M_0 & M_1 & M_2 \end{pmatrix} \quad \hat{M}^{(1)} = \begin{pmatrix} M_1 & M_2 & M_3 & M_0 \\ M_2 & M_3 & M_0 & M_1 \\ M_3 & M_0 & M_1 & M_2 \\ M_0 & M_1 & M_2 & M_3 \end{pmatrix}$$

Figure 2.6 16x16 involution matrices built from the basic building blocks

Finally, the two 16x16 matrices are used to construct the 64x64 block diagonal matrix M' where they are used as the diagonal element in such way: $(\hat{M}^{(0)}, \hat{M}^{(1)}, \hat{M}^{(1)}, \hat{M}^{(0)})$. By applying shift row and matrix multiplication in such way, a full diffusion can be ensured after two iteration of round functions. Furthermore, due to how the matrix is constructed, the number of ones in each row in the 64x64 matrix is three. Therefore, each output bit of the cipher data in the M layer will only be influenced by three input bits. This is to ensure minimum complexity in the M layer to reduce implementation costs.

The key addition operation is actually a 3 inputs XOR operation between the cipher data, the fixed key and the RC-dependent constant. The RC-dependent constants are defined below:

Table 2.2 RC-dependent constant in hexadecimal notation

RC_0	0000000000000000
RC_1	13198a2e03707344
RC_2	a4093822299f31d0
RC_3	082efa98ec4e6c89
RC_4	452821e638d01377
RC_5	be5466cf34e90c6c
RC_6	7ef84f78fd955cb1
RC_7	85840851f1ac43aa
RC_8	c882d32f25323c54
RC_9	64a51195e0e3610d
RC_{10}	d3b5a399ca0c2399
RC_{11}	C0ac29b7c97c50dd

The inverse round functions that executed in the last 5 rounds are similar to the normal round functions. The differences are the substitution and permutation process will be carried out backward. The middle involution and the key additions ensured that the inverse substitution and permutation will not transform the data back into plaintext. Middle involution is the part that connects the forward and inverse round functions. It is composed of shift rows and matrix multiplication of M' similar to the linear diffusion layer. The inverse shift rows will be carried out first, followed by the matrix multiplication and finally another round of shift rows.

2.4 Previous Work

Table 2.3 shows the comparison of design metric including area, latency, and throughput between different lightweight cryptography. The algorithms are fully unrolled to achieved single-cycle design.

Table 2.3 Area, critical path latency and throughput comparison for single cycle lightweight ciphers on ASIC [6].

	Cipher	Area [GE]	Latency [ns]	Throughput [Gbit/s]
32/64	SIMON	8,432.00	29.6	1.00
	SPECK	5,893.25	82.1	0.36
32/80	KATAN	11,939.50	61.2	0.49
	KATAN	24,766.50	75.8	0.79
64/80	PRESENT	22,063.50	39.4	1.51
	RECTANGLE	18,160.75	34.87	1.71
64/128	PRESENT	23,005.75	38.1	1.57
	PRINCE	9,522.75	22.9	2.60
	RECTANGLE	18,935.00	34.68	1.72
	SIMON	23,584.00	41.7	1.43
128/128	SPECK	16,371.00	182.4	0.33
	AES	126,571.00	61.6	1.93

The key performance metric in this research are the area, latency and throughput of the block ciphers. The area was measured in unit of gate count. It was calculated by dividing the total area of the design with the area of 2 input NAND gate. This is a good matric for comparing area as it allows the comparison to be done even in different process node. Meanwhile, the latency was measured in ns while the throughput was measured in giga-bit per second. In overall, the Prince cipher was able to come on top among all the block ciphers in term of latency. Among the ciphers that operate in 64-bit blocks, the Prince was showing the least area utilization, which is comparable even among the smaller block ciphers that run on 32-bit blocks. Due to the low critical path latency, the Prince cipher is able to run at high operating frequency, hence resulting in decent throughput.

This result shows how the loop unrolling technique affect the latency of the design. By applying unrolling on the round-based block ciphers, all the operations in every round are now flatten and placed on the critical path. The effect is terrible

especially for block ciphers with more rounds. The KATAN algorithm with high number of rounds is a good example. Complex round function is another reason that may cause bad latency in unrolled design. The SPECK is one of the algorithms that has a lot of arithmetic operations that do not go well with hardware., thus causing high latency.

2.5 Limitation

The top shortcoming in lightweight cryptography – Prince cipher included, is the level of security that can be achieved by the cipher. The lightweight cryptography typically runs on shorter keys such as 64-bit or 128-bit. The strength of the security is not as good compared to ciphers that utilized 256-bit keys and the lifetime for small keys are generally shorter, mainly due to lesser time required to decipher the keys [7]. Another reason that cause limited security is the simple round functions used. In certain block ciphers, they attempt to overcome the security drawback by introducing more rounds in the algorithm. This could bring negative impact on the performance due to higher latency.

Research was also being carried out to perform third-party analysis of Prince cipher [8]. The author attempted different methods to exploit and break the security of the algorithm. The Prince algorithm utilize the same key throughout the operations, it also has somewhat linear relationship in each round due to the simplistic in the round operations, hence it is totally possible to recover the key used for the encryption via exhaustive manners.

2.6 Research Gap

Despite all the limitations of lightweight cryptography, this research is not focusing on the security of the lightweight cryptography. Reasonable tradeoff in security is acceptable for lightweight cryptography in order to achieve the low-cost, light implementation features, that is suitable for resource-constrained devices. Most

REFERENCES

- [1] J. Han, "Chaining the Secret: Lightweight Authentication for Security in Pervasive Computing," *Eightieth Annual PhD forum on Pervasive Computing and Communications*, 2016.
- [2] M. Knezevic, V. Nikov and P. Rombouts, "Low-Latency Encryption - Is "Lighweight = Light + Wait"?", *International Association for Cryptologic Research*, pp. 426-446, 2012.
- [3] L. Bossuet, G. Gogniat and L. Gaspar, "Architectures of Flexible Symmetric Key Crypto Engines - A Survey: From Hardware Coprocessor to Multi-Crypto-Processor System on Chip," *ACM Computing Surveys*, 2013.
- [4] F. Xinxin, M. Kalikinkar and G. Guang, "A lightweight stream cipher for resource-constrained smart devices.," *Quality, reliability, security and robustness in heterogeneous networks.*, pp. 617-32, 2013.
- [5] J. Borghoff, A. Canteaut, T. Guneysu, E. B. Kavun, M. Knezevic, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, P. Rombouts, S. S. Thomsen and T. Yalcin, "PRINCE - A Low-latency Block Cipher for Pervasive Computing Applications," *Advances in Cryptology -- ASIACRYPT 2012*, pp. 208-255, 2012.
- [6] P. Maene, "Single-Cycle Implementations of Block Ciphers," *Lightweight Roots of Trust for Modern Systems-on-Chip*, pp. 47-64, 2019.
- [7] J. Strombergson, "Some Notes on the Lightweight Block Cipher PRINCE," 2 July 2020. [Online]. Available: <https://www.assured.se/2020/04/24/some-notes-on-the-lightweight-block-cipher-prince/>.
- [8] J. Jean, I. Nikolic, T. Peyrin, L. Wang and S. Wu, "Security Analysis of PRINCE," *In: Fast Software Encryption, FSE 2013*, pp. 92-111, 2014.
- [9] D. Maimut and K. Ouafi, "Lightweight Cryptography for RFID Tags," *Crypto Corner*, pp. 76-79, 2012.
- [10] V. Taraate, *Logic Synthesis and SOC Prototyping*, Singapore: Springer, 2020.

- [11] C. Chen, R. Wei, S. Wang and W. Hu, "Novel Verification Method for Timing Optimization Based on DPSO," *Hindawi*, 2018.
- [12] D. Mills, "RTL Coding Styles That Yield Simulation and Synthesis Mismatches," *SNUG*, 2015.
- [13] Q. Xie, X. Lin, Y. Wang, M. J. Dousli, A. Sharaei, M. Ghasemi-Gol and M. Pedram, "5nm FinFET Standard Cell Library Optimization and Circuit Synthesis in Near- and Super-Threshold Voltage Regimes," *Computer Society Annual Symposium on VLSI*, 2014.