DESIGN METHODOLOGY OF SECURE RFID TAG IMPLEMENTATION

SHUGO MIKAMI

THE UNIVERSITY OF ELECTRO-COMMUNICATIONS

MARCH 2017

DESIGN METHODOLOGY OF SECURE RFID TAG IMPLEMENTATION

SHUGO MIKAMI

THE UNIVERSITY OF ELECTRO-COMMUNICATIONS

GRADUATE SCHOOL OF INFORMATICS AND ENGINEERING

A DISSERTATION SUBMITTED FOR
DOCTOR OF PHILOSOPHY IN ENGINEERING

MARCH 2017

DESIGN METHODOLOGY OF SECURE RFID TAG IMPLEMENTATION

SUPERVISORY COMMITTEE:

CHAIRPERSON: PROFESSOR KAZUO SAKIYAMA

MEMBER: PROFESSOR KAZUO OHTA

MEMBER: PROFESSOR HIROSHI YOSHIURA

MEMBER: PROFESSOR KOICHIRO ISHIBASHI

MEMBER: ASSOCIATE PROFESSOR MITSUGU IWAMOTO

セキュアRFIDタグチップの設計論

三上　修吾

論文概要

本論文ではRFIDタグに着目し，暗号アルゴリズムを用いた認証プロトコルを実行可能な，セキュアなRFIDタグの設計，実装と性能評価を行う．認証プロトコルを実行するためには，暗号アルゴリズムと擬似乱数生成器をRFIDタグに実装することが求められる．セキュアなRFIDタグを実現するため，(A) 認証プロトコルの実装アーキテクチャの決定，(B) 暗号アルゴリズムの選定，(C) 擬似乱数生成手法の確立，(D) RFIDシステムでのシリコンチップの性能評価，に分けて取り組んだ．

本論文では，まず (A) について述べる．認証プロトコルを実行するには，認証の度，暗号アルゴリズムと擬似乱数生成器を複数回動作させる必要がある．そのため，暗号処理にかかる時間がRFIDタグの実装性能に与える影響は大きいと考えられる．これまでの研究では，回路面積が小さいことが求められてきたが，RFIDタグ向けには処理時間の観点から実装アーキテクチャを検討する必要がある．そこで本論文では，認証プロトコルに適した実装アーキテクチャを検討し，処理時間の評価を行う．評価を通して，認証プロトコルで広く利用されるハッシュ関数には，パラレルアーキテクチャが適していることを述べる．

次に，(B) について述べる．これまでの研究で，多数の暗号アルゴリズムが提案されており，暗号アルゴリズムごとに回路単体の実装性能が評価されてきた．しかし，実装性能は評価環境によって異なるため，実装性能の比較評価が困難という問題があった．さらに，RFIDタグ向けのインターフェースを意識した実装はされていなかった．そこで本論文では，RFIDタグ向けのインターフェースを備えた実装を行い，同一の評価環境を用いて暗号アルゴリズムの実装性能の評価を行う．評価結果にもとづき，RFIDタグに適した暗号アルゴリズムについて述べる．

次に，(C) について述べる．RFID タグで擬似乱数を生成する一方策として，(B) の結果を踏まえた擬似乱数生成器の実装が考えられる．一方で，認証プロトコルでは，暗号アルゴリズムと擬似乱数生成器を同時に使用することはないため，暗号アルゴリズムを擬似乱数生成用途にも利用できれば，RFID タグのハードウェアリソースを有効に活用することが可能となる．そこで本論文では，RFID 認証プロトコルで使用されるハッシュ関数を用いた擬似乱数生成手法を提案し，乱数性の評価を行う．評価を通して，軽量な擬似乱数生成手法を確立し，本手法の有効性を述べる．

次に，(D) について述べる．これまでの研究において，暗号アルゴリズムを用いた RFID タグ認証プロトコルは提案されてきた．しかし，認証プロトコルを実行する RFID タグ全体の実装と性能評価はされてこなかった．そこで本論文では，アナログフロントエンドと (A)(B)(C) の結果を含むデジタル処理回路をシングルチップで実装し，RFID タグの実装性能の評価を行う．評価を通して，暗号アルゴリズムを用いた認証プロトコルを実行する，セキュアな RFID タグは実現可能であることを実証する．

DESIGN METHODOLOGY OF SECURE RFID TAG IMPLEMENTATION

SHUGO MIKAMI

ABSTRACT

In this thesis, we focus on radio frequency identification (RFID) tag. We design, implement, and evaluate hardware performance of a secure tag that runs the authentication protocol based on cryptographic algorithms. The cryptographic algorithm and the pseudorandom number generator are required to be implemented in the tag. To realize the secure tag, we tackle the following four steps: (A) decision of hardware architecture for the authentication protocol, (B) selection of the cryptographic algorithm, (C) establishment of a pseudorandom number generating method, and (D) implementation and performance evaluation of a silicon chip on an RFID system.

(A) The cryptographic algorithm and the pseudorandom number generator are repeatedly called for each authentication. Therefore, the impact of the time needed for the cryptographic processes on the hardware performance of the tag can be large. While low-area requirements have been mainly discussed in the previous studies, it is needed to discuss the hardware architecture for the authentication protocol from the viewpoint of the operating time. In this thesis, in order to decide the hardware architecture, we evaluate hardware performance in the sense of the operating time. As a result, the parallel architecture is suitable for hash functions that are widely used for tag authentication protocols.

(B) A lot of cryptographic algorithms have been developed and hardware performance of the algorithms have been evaluated on different conditions. However, as the evaluation results depend on the conditions, it is hard to compare the previous results. In addition, the interface of the cryptographic circuits has not been paid attention. In this thesis, in order to select a cryptographic algorithm, we design the interface of the cryptographic circuits to meet with the tag, and evaluate hardware performance of the circuits on the same condition. As a result, the lightweight hash function SPONGENT-160 achieves well-balanced hardware performance.

(C) Implementation of a pseudorandom number generator based on the performance evaluation results on (B) can be a method to generate pseudorandom number on the tag. On the other hand, as the cryptographic algorithm and the pseudorandom number generator are not used simultaneously on the authentication protocol. Therefore, if the cryptographic circuit could be used for pseudorandom number generation, the hardware resource on the tag can be exploited efficiently. In this thesis, we propose a pseudorandom number generating method using a hash function that is a cryptographic component of the authentication protocol. Through the evaluation of our proposed method, we establish a lightweight pseudorandom number generating method for the tag.

(D) Tag authentication protocols using a cryptographic algorithm have been developed in the previous studies. However, hardware implementation and performance evaluation of a tag, which runs authentication processes, have not been studied. In this thesis, we design and do a single chip implementation of an analog front-end block and a digital processing block including the results on (A), (B), and (C). Then, we evaluate hardware performance of the tag. As a result, we show that a tag, which runs the authentication protocol based on cryptographic algorithms, is feasible.

Contents

Notations

$\|\|$	Concatenation of two bit sequences
$<<< n$	Left cyclic n bit shift
GF(2^n)	Galois field
$+, \oplus$, XOR	EXclusive OR operation
AES	Advanced Encryption Standard
ASIC	Application Specific Integrated Circuit
CMOS	Complementary Metal Oxide Semiconductor
DES	Data Encryption Standard
EEPROM	Electrically Erasable Programmable Read-Only Memory
EPC	Electric Product Code
FF	Flip Flop
FIPS	Federal Information Processing Standards
FPGA	Field Programmable Gate Array
FRAM	Ferroelectric Random Access Memory
GE	Gate Equivalent
HDL	Hardware Description Language
HF	High Frequency
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission

IV	Initial Vector
LF	Low Frequency
LFSR	Linear Feedback Shift Register
LUT	LookUp Tables
MUX	MUltipleXer
NFSR	Nonlinear Feedback Shift Register
NIST	National Institute of Standards and Technology
PRNG	PseudoRandom Number Generator
RF	Radio Frequency
RFID	Radio Frequency IDentification
SHA	Secure Hash Algorithm
SP	Special Publication
SRAM	Static Random Access Memory
UHF	Ultra High Frequency
VCD	Value Change Dump

Part I

Introduction

Chapter 1

Introduction

1.1 Research Background

Internet of Things (IoT) is one of the hottest topics in the field of information technology. IoT is an integration of several technologies to connect the physical world and the cyber world. In IoT system, many different kinds of devices are connected to the Internet. For example, physical objects, such as vehicles, home appliances, and consumer electronics are connected to the Internet. In [67], it is predicted that more than 50 billion devices will be connected to the Internet by 2020. The application areas of these devices are environmental monitoring, weather forecasting, transportation, health care and so on. People can monitor and control the objects from a distance via the Internet. For example, people can monitor and control the status of door safety at home and vehicles remotely. Moreover, smart community and smart energy systems can be realized by IoT systems. From the viewpoint of the economical perspective, both suppliers and consumers take benefit of IoT. The consumers will be able to save money by utilizing smart energy. They will have effective security such as vehicle alarm that informs unauthorized process. The suppliers will have business chance to produce smart devices and technology that increase customer's satisfaction.

In IoT system, Radio Frequency IDentification (RFID) tag is a key device to collect the physical data. The tags are attached to millions of objects such as consumer devices, and books. When the objects, which are used in our daily life, are equipped with tags, the objects are easily identified. In addition, a tag is useful for health monitoring applications.

An RFID system mainly consists of tags and a reader. The tag is a small electric device including an integrated circuit, a memory device, and an antenna. The tag is able to communicate wirelessly with the reader. A usual tag has a unique number and the number is used for identification of the tag. The identification is processed via the wireless communication.

RFID tags are mainly categorized into three types according to the RF, i.e. Low Frequency (LF) tag, High Frequency (HF) tag, and Ultra High Frequency (UHF) tag. LF tag operates at 125 kHz to 134 kHz. HF tag operates at 13.56 MHz. UHF tag operates at 865 MHz to 867 MHz in Europe, and at 902 MHz to 928 MHz in the US. While the range of the UHF band was different according to the countries, the range of UHF band has been unified. For example, in Japan, 920 MHz band is assigned in the UHF band. Table 1.1 shows general tag's caracteristics.

Table 1.1 RF and tag's caracteristics

RF	Low $\longleftrightarrow$ High
Size of antenna	Large $\longleftrightarrow$ Small
Flexibility of antenna	Low $\longleftrightarrow$ High
Maximum read distance	Short $\longleftrightarrow$ Long

According to the report [56], as shown in Table 1.2, RFID market is growing year by year. In

Table 1.2 RFID market

Year	2013	2014	2015	2020
Market ($ billion)	8.8	9.5	10.1	13.2

the report, it is described that the most growing area is UHF RFID tags.

However, there is a concern of counterfeiting of the tag. Since the unique number is constant and the tag sends the unique number to the reader wirelessly, it is easy for an attacker to eavesdrop the tag's response, and to resend it in another time. Therefore, tag authentication is required to prevent the counterfeiting. A widely used authentication protocol is the challenge and response protocol. It is assumed that the tag and the reader share the cryptographic function f and the secret key K. Figure. 1.1 shows the challenge and response protocol.

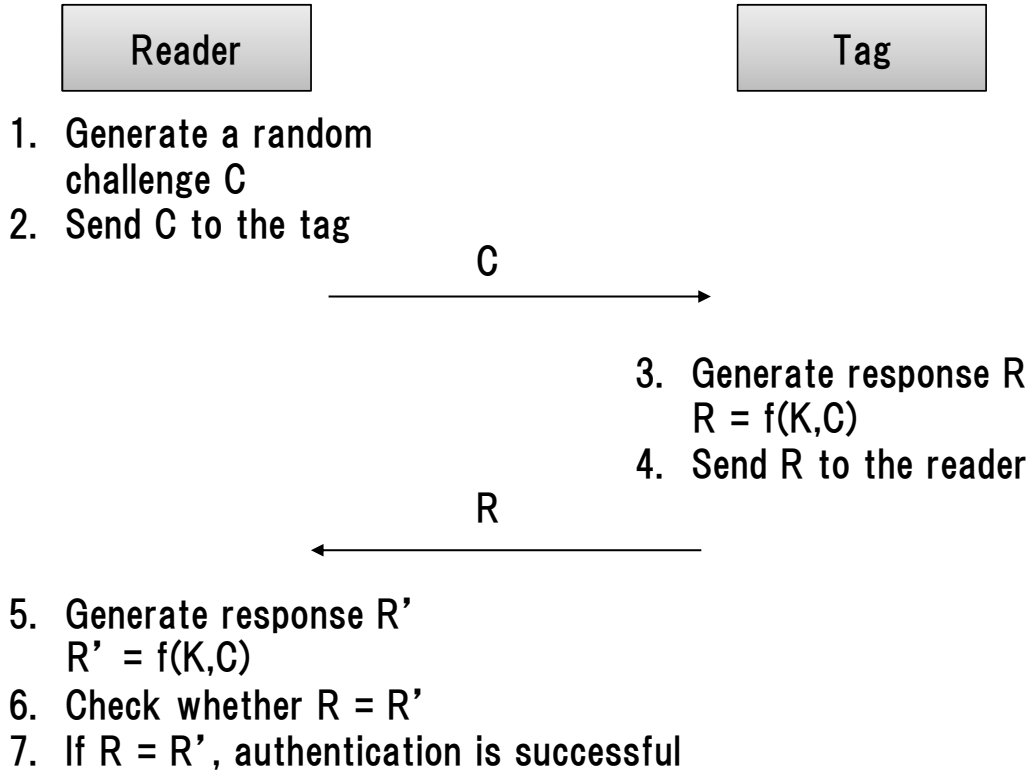


Figure. 1.1 Challenge and response protocol

While the above mentioned protocol removes the risk of the counterfeiting of the tag, the privacy issue is still remained. When an attacker sends the same *C* to the tag repeatedly, the tag returns the same *R*. Therefore, it is easy for the attacker to trace the tag's response by sending the same *C*. Thus, the location of the tag or tag's owner can be easily tracked.

Based on this background, countermeasure techniques against the security and the privacy risks have been studied. One of the techniques is based on Physically Unclonable Functions (PUFs). PUFs exploit inherent physical characteristics of the circuit or the device. Since the output of the PUF is unique, PUF-based RFID tags have been studied. The tag generates a response from the output of the PUF. A merit of the PUF-based RFID tag is that the tag does not hold the secret key in the memory. However, Berker has pointed out that the PUF-based RFID tag, which includes 4-way Arbiter PUF, can be easily attacked using machine learning [11]. The author focuses on a reader that can emulate the tag. The attacker needs to hold the reader to collect the

data communicated between the tag and the reader. Then, the PUF parameter can be reconstructed by software model.

Another technique based on PUFs is PUF-based tag authentication protocol. For example, Aysu *et al.* have proposed a PUF-based tag authentication protocol [5]. The Static Random Access Memory (SRAM) PUF in the tag is used to generate the secret key needed for authentication. To realize the protocol, it is required to implement SRAM PUF, SRAM random number generator, a pseudorandom function, a fuzzy extractor based on BCH error correcting code, and a strong extractor in the tag. The area requirements of the circuit are 3543 Lookup Tables (LUTs) when Xilinx Virtex5 Field Programmable Gate Array (FPGA) is used for performance evaluations. In addition, 2 Mbit SRAM is designated for the PUF. When we assume that a tag runs the challenge and response protocol based on Advanced Encryption Standard (AES), the area requirements of AES are 400 LUTs on Xilinx Virtex5 FPGA [22]. Thus, we deduce that this protocol requires a lot of hardware resources for a tag.

The other technique is tag authentication protocol using a cryptographic algorithm. OMHSO protocol is based on the challenge and response protocol and the hash chain [55]. The secret key is updated with the output of the hash function every time after the challenge and response sequences are succeeded. This protocol consists of cryptographic components, such as a pseudorandom number generator and hash functions. These cryptographic components should be implemented in the tag.

OMHSO protocol

OMHSO protocol is proposed by Hanatani *et al.* in 2011. This protocol, which is an improved version of OSK protocol [88][89], is a mutual authentication protocol between the tag and the server. The desynchronization issue caused by a communication error is solved by the mutual authentication scheme using two secret keys that are the current secret key S_i and previous secret key S_{i-1}. We show the process of the server in Algorithm. 1.1 and the process of the tag in Algorithm. 1.2. S_i and S_{i-1} are shared with the server. X and α are random numbers generated by the server and the tag, respectively. H_0, H_1, and H_2 are hash functions.

Algorithm. 1.1 Authentication process of the server

Input: the current secret key S_i, the previous secret key S_{i-1}, tag's response Y

Output: random number X, server's response Z

1: Generate X.
2: Send X to the tag.
3: Wait until Y arrives to the server.
4: Parse Y as $\alpha||\beta$.
5: Compute $\beta_1 = H_0(S_i, X, \alpha)$.
6: Compute $\beta_2 = H_0(S_{i-1}, X, \alpha)$.
7: **if** $\beta = \beta_1$ **then**
8: Compute $Z = H_1(S_i, X, \alpha)$.
9: **else if** $\beta = \beta_2$ **then**
10: Compute $Z = H_1(S_{i-1}, X, \alpha)$.
11: **else**
12: Compute Z = random number.
13: **end if**
14: Send Z to the tag.
15: **if** $\beta = \beta_1$ **then**
16: Compute $S_{i+1} = H_2(S_i)$.
17: **end if**

Lightweight cryptographic algorithm

At the protocol layer, OMHSO protocol overcomes the security and privacy issue of the tag. However, it is not clear how to realize the tag, which runs OMHSO protocol, at the algorithm layer and at the implementation layer. To tackle this issue, studies from two directions have been performed. One is development of cryptographic algorithm that is suitable for devices with constrained resources.

Traditional cryptographic algorithms have been designed for platforms, which include desktop pc or server, having rich computing resources. However, these cryptographic algorithms are

Algorithm. 1.2 Authentication process of the tag

Input: the current secret key S_i, random number X, server's response Z

Output: tag's response Y

1: Wait until X arrives to the tag.
2: Generate α.
3: Compute $\beta = H_0(S_i, X, \alpha)$.
4: Generate $Y = \alpha||\beta$.
5: Send Y to the server.
6: Wait until Z arrives to the tag.
7: Compute $Z' = H_1(S_i, X, \alpha)$.
8: **if** $Z' = Z$ **then**
9: Compute $S_{i+1} = H_2(S_i)$.
10: **end if**

not suitable for the specific devices with extremely constrained resources. These problems have opened up a research area called lightweight cryptography. Lightweight cryptographic algorithms are designed to be suited for resource constrained environments. eSTREAM project has encouraged developing lightweight stream ciphers [33]. A portfolio profile2 of the project, i.e. Grain has been referred in many studies [53][54][7]. Another portfolio profile2 of the project, i.e. Trivium has been adopted as an ISO/IEC 29192-3 standard [25][26] as well as Enocoro developed by Hitachi, Ltd [50]. Lightweight hash functions, such as SPONGENT [19][20], QUARK [3][6], and PHOTON [48] have been developed and International Organization for Standardization (ISO) standardizing is working.

Implementation of cryptographic components

The other is implementation and evaluation of hardware performance of the cryptographic module or digital processing block that runs a part of the authentication process based on the cryptographic algorithms [51]. In these studies, the digital circuit has been implemented in FPGA or Application Specific Integrated Circuit (ASIC), and hardware performance of the circuit in the sense of area

requirements and power consumption has been evaluated by a simulation.

1.2 Motivation and Contribution of This Thesis

In this thesis, we set basic security strength as 80-bit by the following reason. In the future, Trillion Sensor Society will come and a trillion sensors will be utilized in IoT [101]. We assume that the privacy preserving authentication protocol will be implemented in all the devices. A non-volatile memory is required to realize the privacy preserving authentication protocol. Ferroelectric Random Access Memory (FRAM), which is a kind of non-volatile memory and is now under developing, will be implemented in the device. It is reported that the number of possible write operation will be 10^{12} [38]. When a trillion tags including FRAM memory run the privacy preserving authentication protocol, data patterns needed for device authentication will be $10^{12} \times 10^{12} \sim 2^{80}$.

Towards security and privacy enhancement of the tags, several studies have been done. Unfortunately, the feasibility of the authentication protocol using a prototyping of the tag has not been studied. Namely, a silicon proof of the circuit and performance evaluation of the tag including a digital circuit, an analog front end circuit, and an antenna have not been studied.

We focus on four parts needed for realizing a secure RFID tag. The main contribution in this thesis consists of the following four parts:

1. Design of hardware architecture for cryptographic algorithm
2. Selection of cryptographic algorithm
3. Design of lightweight pseudorandom number generator
4. Design, implementation, and evaluation of secure tag

First, we decide hardware architecture for the authentication protocol. We conclude that the parallel architecture is suitable for hash functions. Second, we select cryptographic algorithms. We evaluate hardware performance of cryptographic algorithms implemented with the above mentioned hardware architecture. We conclude that SPONGENT-160 is a good candidate when low-area requirements and low-power consumption take priority over high-speed performance. On the

other hand, NIST standard hash function Keccak [18] will be used when high-speed performance takes priority over power consumption. Third, we establish a pseudorandom number generating method. We design a lightweight pseudorandom number generator using the hash function that is the cryptographic component of the privacy preserving authentication protocol. We conclude that our proposed method generates a binary sequence that can be thought uniformly distributed. Fourth, we evaluate performance of a silicon chip on an RFID system. We design, implement, and evaluate a tag, which runs the privacy authentication protocol, including the digital processing block, the analog front end block, and the antenna. We conclude that even the standard cryptographic algorithm works in the tag.

First Contribution: Design of hardware architecture for cryptographic algorithm

To implement a cryptographic algorithm in the tag, selection of the cryptographic algorithm is needed since a lot of cryptographic algorithms have been developed. When we select a cryptographic algorithm for the tag, hardware performance of the cryptographic circuit is an important metric. As the hardware performance of the cryptographic circuit depends on implementation architecture of the cryptographic algorithm, it is important to decide the hardware architecture.

Saarinen *et al.* have studied cryptographic algorithms for a tag from a viewpoint of hardware implementation of the cryptographic algorithms [96]. They have mentioned that previous works had been focused on minimizing the area requirements of the cryptographic algorithms and had not pay attention to the time needed for the cryptographic process. They mentioned that the lightweight hash functions are inappropriate for a tag since these algorithms take a large number of clock cycles.

However, they have not taken care of the security strength. They have evaluated the number of clock cycles needed for generating 16-bit output. We deduce that this data length is too short to enhance the security of the tag. This is because the outputs of two different tags have the same value with a probability of $1/2^8$. This situation causes authentication failure of the tags.

The first contribution of this thesis is that we decide the hardware architecture of the cryptographic algorithms for a tag. We have focused on hash functions and stream ciphers that are widely used for a cryptographic component of the authentication protocols. Hardware perfor-

mance of the cryptographic algorithms in the sense of time needed for the cryptographic process depends on hardware architecture. Thus, we have picked up and designed hardware architectures for the cryptographic algorithms. One is the parallel architecture. All bits of data stored in the Flip Flops (FFs) are taken as the input to the update function and the output of the update function are returned to the FFs. A characteristic of the parallel architecture is that all bits of data stored in the FFs are updated simultaneously. The other is the sequential architecture. A part of the FFs are taken as the input to the update function and the output of the update function is stored in a buffer. Then, the data stored in the FFs shift sequentially. Finally, the data stored in the buffer are returned to the FFs. A characteristic of the sequential architecture is that data stored in the FFs are updated sequentially.

We have evaluated the number of clock cycles of the cryptographic circuits based on these hardware architectures. Our evaluation results show that when the hash function is implemented with the parallel architecture, the numbers of clock cycles are less than 1600. On the other hand, when the hash function is implemented with the sequential architecture, the numbers of clock cycles reach to 900000. Thus, the number of clock cycles differ 600 times according to the hardware architecture. The numbers of clock cycles of the stream ciphers are less than 1600 when these algorithms are implemented with both architectures.

In conclusion, we deduce that the parallel architecture is suitable for the hash function and both of the architectures are suitable for the stream ciphers to achieve high speed performance.

Second Contribution: Selection of cryptographic algorithm

Usually, hardware performance of a cryptographic algorithm has been reported in a paper that proposes the cryptographic algorithm. In these studies, different tools are used for performance evaluations. However, hardware performance of the cryptographic algorithms depend not only hardware architecture but also evaluation tools such as evaluation platform, compiler, or libraries. Therefore it is difficult to compare hardware performance of the cryptographic algorithm simply using these evaluation results.

Kerckhoff *et al.* have evaluated hardware performance of 6 block ciphers in the sense of area requirements, power consumption, throughput, and energy using the same evaluation tool [58].

Batina *et al.* have evaluated hardware performance of 11 block ciphers in the sense of area requirements, power consumption, and energy using the same evaluation tool [16]. These studies have focused on a fair comparison between AES and recently developed block ciphers.

However, in these studies, the interface of the cryptographic circuit has not been paid attention. The cryptographic circuit is connected to another circuit in the tag to realize some function. Thus, we deduce that it is important to evaluate hardware performance of the cryptographic algorithms under the condition that the interface of the cryptographic circuit is designed to meet with the tag.

The second contribution of this thesis is that we select a cryptographic algorithm for a tag based on hardware evaluation results. We focus on hash functions and stream ciphers that are vastly used for tag authentication protocols. First, we design the interface of the cryptographic circuit. As shown in Algorithm. 1.2, a part of the input data for the cryptographic circuit is sent from the reader. The output of the cryptographic circuit are compared with data received from the reader. This comparison process is done in another circuit. Therefore, it is natural to store the input data and the output data of the cryptographic circuit in a temporal memory that can be accessed from various circuits. Thus, we design the interface of the cryptographic circuit based on SRAM. The data width of the SRAM has variation such as 8-bit, 16-bit, and 32-bit. As the data width is larger, the area requirements of the circuits are larger. To achieve low-area requirements, we select the data width to 8-bit. We design the interface of the cryptographic circuit based on 8-bit SRAM.

Second, we implement the hash functions and the stream ciphers in ASIC based on the hardware architecture that we decided before. Namely, we implement the hash functions with the parallel architecture and the stream ciphers with both the parallel architecture and the sequential architecture. Then, we evaluate hardware performance of the cryptographic algorithms in the sense of area requirements and the number of signal toggles. The number of signal toggles is a key metric for power consumption. The power consists of two parts. One is the static power resulted from the leakage current. The other is the dynamic power resulted from the switching activity of the transistors. As the static power is small in Complementary Metal Oxide Semiconductor (CMOS) devices, the dynamic power takes a dominant part of the power consumption. The number of signal toggles reflects the switching activity of the transistors directly. From our evaluation results,

SPONGENT-160 achieves low-area requirements and low-power consumption. Keccak achieves high-speed performance while it takes large area requirements and consumes power.

In conclusion, when low-area requirements and low-power consumption have priority over high-speed performance, SPONGENT-160 is a good candidate for the hash function on the tag. On the other hand, when high-speed performance has priority over low-area requirements and low-power consumption, Keccak can be used as the hash function for the tag. When the stream ciphers are required, Grain and Enocoro implemented with the parallel architecture or the sequential architecture can be used.

Third Contribution: Design of lightweight pseudorandom number generator

The tag authentication protocol consists of two cryptographic components; hash function and PseudoRandom Number Generator (PRNG). Previous studies make it cleared that security weakness of tags caused from mainly pseudorandom number generator [68][45][78][49][44]. Therefore, PRNGs for a tag have been studied[65][60].

Lopez *et al.* have focused on Blum-Blum-Shub (BBS) PRNG [65]. Katti *et al.* have proposed PRNG based on linear congruent generator using addition, multiplication, and modular reduction [60]. A lightweight PRNG using addition and multiplication has been developed [71]. Mujahid *et al.* have proposed a PRNG using bitwise right rotation, bitwise OR, multiplication, and modulo addition [69]. M-Segui *et al.* have proposed PRNG based on Linear Feedback Shift Register (LRSR) configured with multiple polynomials selected by a physical source [94][95]. Chen *et al.* have proposed multiple polynomial LFSR based PRNG [27].

However, from the viewpoint of implementation, it is required to implement not only the cryptographic circuit but also a PRNG circuit. Therefore, to achieve low-area requirements of the tag, pseudorandom number is desired to be generated using the cryptographic circuit since the cryptographic circuit and PRNG circuit are not activated simultaneously. While pseudorandom number generating methods using cryptographic algorithms are standardized by National Institute of Standards and Technology (NIST) [79], these methods require much hardware resources.

Therefore, we focus on a hash function and we study a lightweight pseudorandom number generating method using the hash function. First, we propose a lightweight pseudorandom number

generating method based on hash chain. Our method works as follows. The seed is taken as the input to the hash function. Then, the output of the hash function is divided into two parts. One is used for the updated seed, which is stored secretly, and the other is used for the pseudorandom number that is returned.

Second, we evaluate the lightweight pseudorandom number generating method from the viewpoint of randomness. We use NIST test suite [81] and Diehard test suite [30], since these tests are widely used as a statistical test to measure the randomness of the data. We select SPONGENT-160 as the hash function since it has been cleared from our studies that the algorithm is a good candidate for a tag. We generate a binary sequence using our proposed method and evaluate the randomness of the binary sequence with NIST test suit and Diehard test suite.

Our evaluation results show that the binary sequence passes all tests included in NIST test suite and Diehard test suite. Thus, we deduce that the output of our proposed method can be thought uniformly distributed. Our method achieves 53 % less area requirements of the cryptographic circuit compared with the area requirements of a cryptographic circuit including a lightweight hash function and a lightweight stream cipher as PRNG.

To realize the NIST method, additional components such as an expansion function are required to generate a long sequence of pseudorandom bits. On the other hand, our proposed method does not require additional components such as an expansion function, as the bit length of the pseudorandom number is not long for RFID tag. Therefore, our proposed method achieves low area requirements. In conclusion, our lightweight pseudorandom number generating method is useful for tag.

Fourth Contribution: Design, implementation, and evaluation of secure tag

The hardware performance of the cryptographic modules or the digital processing blocks that run a part of the authentication process has been studied. FPGA implementation results of the digital processing block that executes a hash-based challenge-response protocol have been studied [105]. Implementation results of OSK protocol on an IC card have been studied [1]. ASIC implementation results of the digital processing block including AES [80] and Grain [7][53][54] have been studied [36]. ASIC implementation results of CRYPTOGPS including PRESENT [21] have been

studied [91].

However, as an application of the cryptography, the cryptographic circuit has not been combined with a hardware restricted device. The feasibility of the protocol has not been studied. Namely, implementation and evaluation of the whole tag has not been studied. To realize a tag that runs the authentication protocol, not only a digital processing block but also several blocks such as an analog front end, non-volatile memory, and an antenna are required to be implemented in the tag. As these blocks operate in a mutually coupled manner to run the authentication protocol, it is important to implement these blocks in the tag for performance evaluation.

In addition, it has been widely believed that a lightweight cryptographic algorithm is an essential component for a tag and a standard cryptographic algorithm is not suitable for a tag. However, performance overhead of a tag resulted from the standard cryptographic algorithm compared with the lightweight cryptographic algorithm has not been cleared. Therefore, a field test is required.

We design a completely assembled tag including a digital processing block, an analog front end block, Electrically Erasable Programmable Read-Only Memory (EEPROM), SRAM, and an antenna. To achieve low area requirements of the tag, we implement these blocks except the antenna in a single chip. As a cryptographic algorithm, we select SPONGENT-160 and we implement the algorithm with the parallel architecture in accordance with our performance evaluation results. To discuss the impact of the hardware performance of the cryptographic algorithm for the tag, we also select Keccak and we implement the algorithm with the parallel architecture. To generate pseudorandom number, we use SPONGENT-160 or Keccak as the instance of the hash function needed for our proposed lightweight pseudorandom number generating method. These two hash functions can be switched and used by introducing a debug functionality. Namely, when SPONGENT-160 is used as the hash function, SPONGENT-160 is called for PRNG. When Keccak is used as the hash function, Keccak is called for PRNG.

We carried out a full silicon proof of the tag. Then, we evaluate hardware performance of the tag in the sense of area requirements, power consumption, operating time, and maximum read distance of the tag.

Our evaluation results show that when SPONGENT-160 is used, the maximum read distance is 10 cm. The operating time is less than 30 ms. The area requirements of the cryptographic block

are 10 kGE. The average power consumption of the cryptographic block is 260 μW when the hash function is active. When Keccak is used, the maximum read distance is 8.5 cm. The operating time is less than 20 ms. The area requirements of the cryptographic block are about 39.4 kGE. The average power consumption of the cryptographic block is 536 μW when the hash function is active.

We have designed a passive tag with the analog power block that has enough capacity to supply power to the circuit absolutely. When we use smaller analog power block, the impact of the cryptographic block in the sense of power consumption will become larger. Then the lightweight cryptographic algorithm will have advantage.

In conclusion, the feasibility of the privacy preserving authentication protocol based on cryptographic algorithms is demonstrated. In addition, it is verified that the standard cryptographic algorithm works on the tag.

1.3 Towards Future Research Topic

As a usual tag sends a constant data to a reader, there is a privacy concern that an owner of the tag can be tracked by tracing the tag's response. To overcome the privacy issue, the privacy-preserving tag authentication protocol based on a cryptographic algorithm has been developed. Our results described in this thesis show the feasibility of the authentication protocol for the tag. Thus, the security and the privacy of the tag can be enhanced by running the protocol sequence in the tag. On the other hand, since the tags are attached to many kinds of devices, an attacker can handle the tags easily. Therefore, the tag will be exposed to side channel attack.

Countermeasure for side channel attack

We deduce that the privacy preserving authentication protocol is vulnerable to side channel attack. After the tag receives a random number from the reader and the tag generates a random number, the tag generates a response with a hash function by inputting the secret key and two kinds of random numbers. Then, the tag sends the response to the reader. Immediately after this step, we move away the tag from the reader. The tag cannot receive enough power to run the protocol

sequence from the reader. Since the secret key in the tag is not updated, the secret key will be taken as the input to the hash function to generate tag's response at the next time. We deduce that the secret key in the tag can be revealed by differential side channel attack. Since we implement the cryptographic block and the analog power block in a single chip, it seems difficult to measure the power consumption of the cryptographic block by tapping directly the power supply line from the analog power block to the cryptographic block. On the other hand, since electro magnetic wave resulted from an electric current in the cryptographic block can be captured by EM probe, electro magnetic analysis can be applied to the tag. Therefore, we deduce that the secret key in the tag can be revealed by differential electro magnetic analysis.

Our previous results show that the secret key of Enocoro implemented in FPGA can be revealed by differential power analysis using 1000 traces [75]. It takes about 5 minutes to collect the traces. When the differential electro magnetic analysis is applied to the tag, it is needed to move the tag and to initialize the tag's state. It takes about 100 ms to volatile data stored in SRAM [24]. Power-on off operations of the reader is also needed. Therefore, it is predicted that it takes about 30 minutes to collect traces for the differential electro magnetic attack.

As the countermeasure techniques against the differential side channel attack, data hiding method and data masking method have been studied [70]. The data hiding method includes WDDL [102]. It has been cleared that information leakage occurs even after applying the data hiding method [70][98][74]. This information leakage results from a balance of load capacitance or delay time of input signals. The data masking method includes threshold implementation technique [85]. The threshold implementation technique is based on masking using random number and secret sharing. According to our previous results, the area requirements of the threshold implementation version of Enocoro are three times larger than that of unprotected version of Enocoro [75]. The linear components in the cryptographic circuit are implemented in three parallel manners. The number of input and output signals of the nonlinear components in the cryptographic circuits are three times larger than that of unprotected version of the nonlinear components. Therefore, it is predicted that the number of signal toggles in the threshold version of the cryptographic algorithm is three times larger than that of unprotected version of the algorithm. According to our evaluation results in this thesis, there is a liner relationship between the number

of signal toggles and the power consumption. Therefore, the threshold implementation version of the cryptographic circuit will consume three times larger power than unprotected version of the one.

1.4 Structure of This Thesis

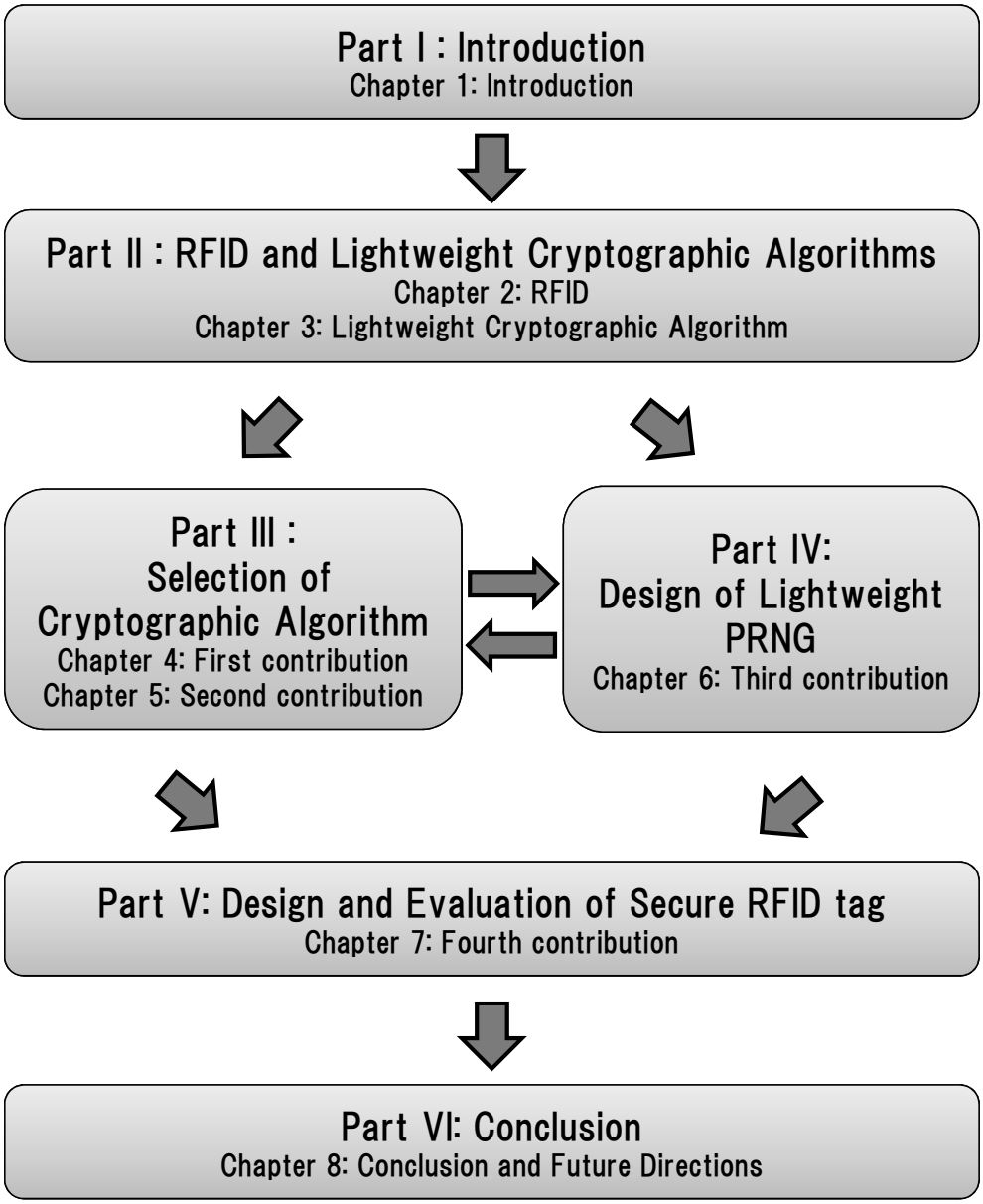


Figure. 1.2 Structure of this thesis

Figure. 1.2 shows the structure of this thesis. In Part I, we describe the introduction including research background, motivation, and contribution of this thesis. In Part II, we summarize the knowledge of RFID system including tag, RFID standards, RFID reader, backend system, security threat, and requirement. Further, we describe the knowledge of cryptographic algorithms, hardware implementation platform, and performance evaluation metrics. In Part III, we describe selection method of cryptographic component including hardware architecture and performance evaluation results of cryptographic algorithms. In Part IV, we propose a lightweight pseudorandom number generating method using a hash function and we evaluate the randomness of the output of the proposed method. In Part V, we describe a secure RFID tag including the specification of the tag and hardware evaluation results of the tag. Finally, we conclude this thesis and suggest future research directions in Part VI.

Part II

RFID and Lightweight Cryptographic Algorithms

Chapter 2

RFID

2.1 Introduction

In this chapter, we summarize RFID. We describe variation of RFID tags, RFID standards, RFID reader, backend system, security, and privacy issue.

2.2 Tags

RFID tags are categorized into three types according to the RF [37].

2.2.1 LF Tag

Low Frequency (LF) band covers the RF spectrum ranging from 125 kHz to 134 kHz. As this RF is lower than the others, the size and the number of turns of the antenna are larger than that of the others. In addition, the order of the read rate is a few bits/s. The typical operating distance of the tag is around 1 m. The application areas of the tag are vehicle immobilizers, pet identification, and access control.

2.2.2 HF Tag

High Frequency (HF) tag operates at 13.56 MHz. As the tag operates at high frequency, the tag achieves higher data rates than that of the LF tag. The order of the data rate is Kbits/s. As the HF tag operates in shorter wavelengths compared to the LF tag, the antenna size is smaller than that of the LF tag. Thanks to this property, the antenna takes flexible form such as a paper label, and a card. The typical operating distance is about 0.7 m. The application areas of the tag are credit cards, ticketing, and library management.

2.2.3 UHF Tag

Ultra High Frequency (UHF) tag covers the RF spectrum ranging from 868 MHz to 928 MHz. The frequency band of the UHF tag are different in the countries. For example, in the US, the frequency band is 902-928 MHz. In Europe, the frequency band is 865-867 MHz. In Japan, the frequency band is 920 MHz. For energy propagation, the tag reflects back the electromagnetic wave that received from a reader. This technique is called backscatter modulation. The dipole antennas are typically used in the UHF tag. The form of the antenna can be linear or circular depending on the application of the tag. The typical operating distance is from 1 to 6 m. The applications of the tag are the supply chain, the logistics, and the item management.

2.3 RFID Standards

2.3.1 EPCGlobal

There are two important standards. One is developed by EPCGlobal. EPCGlobal is a consortium of several companies and universities. EPCGlobal Generation 2 is the most important standard developed by EPCGlobal [34]. In EPCGlobal Geneation 2, tags are classified into 5 classes. Class 0 tag refers to the read only passive tag. ID is implemented in the tag when the integrated circuit is manufactured. Class 1 tag refers to write-once passive tag. ID of this class of the tag can be written once by users. Class 2 tag refers to rewritable passive tag. Class 2 tag has rewritable

memory and authenticated access control. Class 3 tag refers to semi active tag that has a battery to supply power. Finally, class 4 tag refers to active tag. The communication distance of the tag is longer and complex processes can be done on the tag. EPC Tag Data Standard Version 1.6 [35] defines the ID length as 96, 113, 170, 195, 198, and 202-bit.

2.3.2 ISO

The other standard is developed by ISO. ISO standards define RFID in ISO 18000. ISO 18000 is a multistandard that deals with operating frequencies. ISO 18000-1 standard describes reference architecture and definition of parameters. ISO 18000-2 refers to parameters for air interface communications below 135 kHz. ISO 18000-3 refers to parameters for air interface communications at 13.56 MHz. ISO 18000-4 refers to parameters for air interface communications at 2.45 GHz. ISO 18000-5 is not listed. ISO 18000-6 refers to parameters for air interface communications at 860 MHz to 960 MHz. ISO 18000-7 refers to parameters for air interface communications at 433 MHz. Here, ISO18000-6 standard is met with EPCGlobal Generation 2 Class 1.

Another important ISO standards are ISO 14443 and ISO 15693. ISO 14443 standard defines the contactless integrated circuits cards (proximity cards). ISO14443-1 standard refers to physical characteristics. ISO14443-2 standard refers to the RF power and signal interface. ISO 14443-3 standard refers to initialization and anticollision. ISO 15963 standards define the contactless integrated circuits cards (vicinity cards). ISO/IEC 15963 defines the ID length as 64-bit.

2.4 RFID Reader

RFID reader is a device that sends data to the tag and receives data from the tag. The reader consists of an antenna, a microprocessor, and an interface for sending data to the backend system. The antenna is required to communicate with the tag wirelessly. The microprocessor is used to control the device. The reader sends the data received from the tag to a backend system. The reader generates the electromagnetic field to supply power to the tag.

2.5 Backend System

The backend system is connected to the reader. The backend system consists of a server. The backend system stores tag's data including ID and processes data that are sent to the tag. The backend system controls the tags.

2.6 Security and Privacy

2.6.1 Threat

A tag has risks against security and user privacy. As the tags are inexpensive devices and the tags communicate wirelessly, RFID system has vulnerabilities and is susceptible to malicious attacks. An attacker tries to imitate a legitimate tag to communicate with the reader. The attacker uses an electronic device having an antenna to capture the data, which includes tag's ID stored in the memory of the legitimate tag, communicated between the legitimate tag and the reader. The attacker copies the data. A cloned tag is easily manufactured by copying the data of the legal tag into the memory. The attacker is able to fool the reader to accept the cloned tag as a legitimate tag. It is clear that two tags with the same identifier in a single RFID system will confuse the system and compromise the security of the system.

In addition, the captured data are used to execute replay attack by the attacker. The attacker may capture valid communication data and resend the data to the reader at a later time. Even if the tag sends an encrypted data to the reader, the tag is still vulnerable to the replay attack because the tag sends a fixed data.

As the tags are inexpensive devices, tamper resistant devices are no required. Therefore, the tags may suffer from side channel attacks that attempt to reveal secret data including keys stored in the memory of the tag. A circuit shows different characteristics of power consumption and electromagnetic wave radiation depending on data that are processed in the circuit. An attacker records the power consumed by the tag or the electromagnetic wave generated by the tag. Then, the attacker amplifies the difference of the power or the electromagnetic wave depending on the

data processed in the tag to recover the secret data stored in the tag.

As the tags are small and embedded into the objects, it is hard for human beings to identify the tags. However, the tag sends response to the readers whenever the tags receive queries from the reader. Therefore, anyone who has a reader can read the tag's response. The data sent from the tags provide identification data or location data of the tags. Information leakage about the tag and the object, which is attached with the tag, is called contents privacy issue. The identification data may include data about the objects such as price, name, and date when the user buys the objects. When the tags are attached to particular objects, the tag's response will include sensitive information of the users. For example, when the tag is attached to the medicine, the tag's response will reveal a particular disease. When the tag is attached to books, the tag's response will link to thought.

Since the tag's response stays the same, it is easy for an attacker to track the user of the tag by tracing the tag's response. Information leakage about location of the tag and users is called location privacy issue.

2.6.2 Requirement

To prevent incoming an illegal tag into the RFID system, tag authentication is required. To solve the contents privacy issue, it is effective to encrypt tag's response. This is because an attacker cannot understand the meaning of the data. Even if the tag's response is encrypted, the attacker is able to trace the tag since the encrypted data are fixed. To solve the location privacy issues, the tag's response should be variable.

The secret information in the memory needed to generate tag's response should not be leaked to the attacker. Once the secret information is revealed by the attacker, the attacker can recover the tag's previous responses. This means that the attacker is able to traceback the action of the users. On the other hand, a tamper resistant memory is hesitated to use in the tag as the tag is an expensive device. To enhance the security of the tag against this issue, forward security is required. Forward security means that even if the current secret information is leaked to the attacker, it is difficult to recover the previous tag's data.

Chapter 3

Lightweight Cryptographic Algorithms

3.1 Introduction

In this chapter, we summarize cryptographic components that are key components to enhance security and privacy of the tag. We describe specification of some lightweight cryptographic algorithms, hardware implementation platforms, and hardware performance evaluation metrics.

3.2 Symmetric Cryptographic Primitives

Symmetric cryptographic primitive uses the same cryptographic keys for both encryption of the plaintext and decryption of the ciphertext. The keys must be shared between two or more parties that send and receive data. In addition, the keys must be distributed to the parties securely. For example, the legitimate parties sends the keys over a secure channel. There are three classes of the symmetric cryptographic primitives: block ciphers, stream ciphers, and hash functions.

3.2.1 Block Cipher

A block cipher is a symmetric key algorithm operating on fixed length of bits, called block. Usually, the block lengths are set to 64-bit, 80-bit, or 128-bit. A block cipher transforms a block of a plaintext into a block of a ciphertext that has the same block length with the plaintext. Most of the block ciphers iterate the same function, which is called a round function, several times. The round key, which is taken as an input to the round function, is varied in each round. Decryption is the inverse operation of the encryption. It recovers the plaintext from the ciphertext using the same secret key.

A lot of block cipher algorithms have been developed. DES algorithm had been a NIST standard block cipher listed in NIST FIPS 46. The key size is 56-bit and the block size is 80-bit. However, as an effective attack on DES using a cryptanalysis technique called linear cryptanalysis has developed, NIST withdraw NIST FIPS 46 [84]. DES has been replaced by AES that has been standardized by NIST in 2001 [80]. The key sizes are 128-bit, 192-bit, and 256-bit. The block size is 128-bit. Other standardized block ciphers are Camellia [8], CREFIA [97], and PRESENT [21]. Camellia has been adopted as an ISO/IEC 18033-3 standard. CREFIA and PRESENT have been adopted as ISO/IEC 19291-2 standard.

3.2.2 Stream Cipher

A stream cipher is a symmetric cryptographic algorithms operating on flexible length of bits sequentially in the unit of bit or byte. A stream cipher consists of two processes. One is a pseudorandom number generating process and the other is an encryption process. A pseudorandom number generating process generates a keystream from a seed that is treated as a secret key. The keystream is combined with a plaintext using XOR operation on the encryption process. Decryption is the same operation of the encryption. It recovers the plaintext from the ciphertext using the same seed.

A lot of stream ciphers have been developed. MUGI [104], SNOW 2.0 [32], Rabbit [23], Decim v2 [12], and KCipher-2 [61] have been adopted as ISO/IEC 18033-4 standard. Enocoro

and Trivium have been adopted as ISO/IEC 29192-3 standard.

3.2.3 Hash Function

A hash function maps binary strings of arbitrary length to binary strings of a fixed length. The input data is called message and the output is called hash value. Hash functions have many applications such as digital signature, message authentication code, authentication, fingerprinting, and checksum.

A lot of hash functions have been developed. The most widely used hash functions have been SHA-1 and MD5. These hash functions output 128-bit hash value. However, the security of MD5 is compromised and an attack on SHA-1 was reported in 2005.

The improved versions of SHA-1 are called SHA-2 and they are standardized by NIST as NIST FIPS 180-4 [83]. SHA-2 is a set of hash functions including SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, and SHA-512/256. The output of these hash functions are 224-bit, 256-bit, 384-bit, 512-bit, 224-bit, and 256-bit, respectively. SHA-512/224 and SHA-512/256 are truncate versions of SHA-512.

In 2007, NIST opened a competition to select a new standard hash function SHA-3. 51 kinds of hash functions have been submitted to the competition. In 2009, 14 kinds of hash functions have been selected as second round candidates. The second round candidates are Keccak [18][17], BLAKE [4], Grϕstl [47], JH [103], Skein [40], Blue midnight wish [46], CubeHash [10], ECHO [13], Fugue [52], Hamsi [57], Luffa [29], Shabal [14], SHAvite-3 [15], and SIMD [63]. In 2010, NIST announced 5 finalists including Keccak, BLAKE, Grϕstl, JH, and Skein. In 2012, Keccak has been selected as SHA-3.

Keccak

Keccak is a family of hash functions and it has been developed by Bertoni *et al.* Keccak is based on a sponge construction and has seven variations depending on the size of the internal state. The permutation function is called Keccak-f. Keccak-f is characterized by the width b of the permutation. The width b is selected from {25, 50, 100, 200, 400, 800, 1600} and is equal to the width of the state. The state is represented with three dimensional matrix. The size of the matrix is 5-bit $\times$ 5-bit $\times$ w-bit, where $w = b/25$. The two dimensional matrix $5 \times 5 \times 1$ is called a slice.

Algorithm. 3.1 Keccak algorithm

Input: the state A, the round constants RC

Output: the updated state A

1: **for** i = 1, 2, . . . ,24 **do**
2: **for** x = 0, 1, . . . ,4 **do**
3: C[x] = A[x,0] ⊕ A[x,1]⊕ A[x,2]⊕ A[x,3]⊕ A[x,4].
4: **end for**
5: **for** x = 0, 1, . . . ,4 **do**
6: D[x] = C[x-1] ⊕ ROT(C[x+1],1).
7: **end for**
8: **for all** (x,y) such that $0 \leq x \leq 4, 0 \leq y \leq 4$ **do**
9: A[x,y] = A[x,y] ⊕ D[x].
10: **end for**
11: **for all** (x,y) such that $0 \leq x \leq 4, 0 \leq y \leq 4$ **do**
12: B[y,2x+3y] = ROT(A[x,y], r[x,y]).
13: **end for**
14: **for all** (x,y) such that $0 \leq x \leq 4, 0 \leq y \leq 4$ **do**
15: A[x,y] =B[x,y] ⊕ ((not B[x+1,y]) and B[x+2,y]).
16: **end for**
17: A[0,0] = A[0,0] ⊕ RoundConst.
18: **end for**

The one dimentional matrix $1 \times 1 \times w$ is called a lane.

The state is initialized with 0. The input message is padded with a single 1 and then zeros are added until the message length is a multiple of the rate size r. Then, the message is cut into blocks that have r bits. The block is xored with the first r bits of the state interleaved with the application of the permutation. The state is taken as the input to the permutation and is updated with the output of the permutation.

After all blocks are xored with the state, the squeezing process begins. In the squeezing process,

the first r bits of the state are returned as a hash block interleaved with the application of the permutation. The state is taken as the input to the permutation and is updated with the output of the permutation. The squeezing process continues until desired bits of hash values are generated. The permutation consists of the round function. The round function consists of five functions. Namely, θ function, ρ function, π function, χ function, and ι function. The round number is defined by $n_r = 12 + 2 \times l$, where $2l = w$. We describe a pseudo code of Keccak-f in Algorithm. 3.1. B[x,y], C[x,y], and D[x] are intermediate values. r[x,y] is constant value.

3.3 Lightweight Cryptographic Algorithms

In this section, we describe specification of some lightweight cryptographic algorithms. Table 3.1 shows a list of the lightweight cryptographic algorithms that we treat in this thesis. Trivium and

Table 3.1 List of the lightweight cryptographic algorithms

Lightweight stream cipher	Grain
	Trivium
	Enocoro
Lightweight hash function	SPONGENT
	QUARK
	PHOTON

Enocoro are ISO standard lightweight stream ciphers. These algorithms are listed in ISO/IEC 29192-3. Trivium is a finalist of the eSTREAM project, which was started by ECRYPT to select stream ciphers, in Profile 2. Grain is another finalist of the project.

ISO/IEC has started discussion of standardizing lightweight hash functions. Several kinds of lightweight hash functions will be adopted as ISO/IEC standard lightweight hash function and will be listed in ISO/IEC 29192-5. As the candidates of the lightweight hash functions, SPONGENT, QUARK, and PHOTON have been developed.

3.3.1 Grain

Grain is designed by Hell, Johansson, and Meier. It has been submitted to eSTREAM in 2004 and it has been selected as an eSTREAM Portfolio Profile 2 [54]. The key size is 80-bit and the IV size is 64-bit. Grain is a bit-oriented synchronous stream cipher. Namely, the keystream is generated independently from the plaintext. The size of the internal state is 160-bit.

The algorithm consists of two phases. The first one is the initialization phase. In this phase, key and IV are taken as the input to the internal state. The internal state is updated 160 times repeatedly. The second one is the pseudorandom number generating phase. In this phase, the internal state is continuously updated and the keystream is generated interleaved with the update of the internal state.

The internal state consists of two shift registers. One is 80-bit LFSR and the other is 80-bit NFSR. LFSR is represented as $s_i, s_{i+1}, \ldots s_{i+79}$ from the leftmost bit. NFSR is represented as $b_i, b_{i+1}, \ldots, b_{i+79}$ from the leftmost bit.

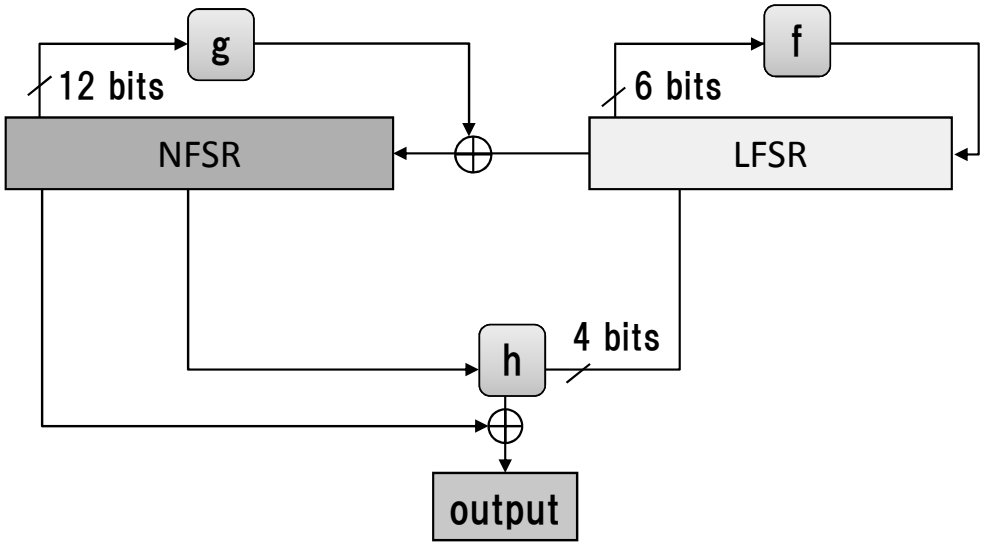


Figure. 3.1 Structure of Grain

The key is taken as the input to NFSR. Figure. 3.1 shows a structure of Grain. The IV and

constant number are taken as the input to LFSR. The output of LFSR is taken as an input to NFSR. Here, $f(s)$ and $g(s,b)$ are update functions for LFSR and NFSR, respectively. $h(s,b)$ is output function. The output of $h(s,b)$ is feedbacked to LFSR in the initialization phase. In the pseudorandom number generating phase, the output of $h(s,b)$ is returned as keystream.

Since both LFSR and NFSR shift one bit per clock, the algorithm generates keystream one bit per clock. However, it is easy to speed up by implementing the update functions and the output function t times parallely and shifting t bits per clock. The number of clock cycles needed for the initialization phase is $160/t$. The cipher generates t-bit keystream per clock. The elements in the functions are required to be implemented t times. By implementing these functions parallely, it is possible to speed up the cipher without lengthening the critical path. Since 16 bits of the state are not taken as an input to the functions, up to 16 iterations can be computed at a clock.

(1) Update function

The update function of LFSR, $f(s)$ is defined by the following equation.

$$f(s) = s_{i+62} + s_{i+51} + s_{i+38} + s_{i+23} + s_{i+13} + s_i. \tag{3.1}$$

The update function of NFSR, $g(s,b)$ is defined by the following equation.

$$\begin{aligned} g(s,b) = {} & s_i + b_{i+63} + b_{i+60} + b_{i+52} + b_{i+45} + b_{i+37} + b_{i+33} + b_{i+28} + b_{i+21} + b_{i+15} + b_{i+9} \\ & + b_i + b_{i+63}b_{i+60} + b_{i+37}b_{i+33} + b_{i+15}b_{i+9} + b_{i+60}b_{i+52}b_{i+45} + b_{i+33}b_{i+28}b_{i+21} \\ & + b_{i+63}b_{i+45}b_{i+28}b_{i+9} + b_{i+60}b_{i+52}b_{i+37}b_{i+33} + b_{i+63}b_{i+60}b_{i+21}b_{i+15} \\ & + b_{i+33}b_{i+28}b_{i+21}b_{i+15}b_{i+9} + b_{i+52}b_{i+45}b_{i+37}b_{i+33}b_{i+28}b_{i+21}. \end{aligned} \tag{3.2}$$

(2) Output function

$h(s,b)$ is defined by the following equation.

$$\begin{aligned} h(s,b) = {} & s_{i+25} + b_{i+63} + s_{i+3}s_{i+64} + s_{i+46}s_{i+64} + s_{i+64}b_{i+63} + s_{i+3}s_{i+25}s_{i+46} + s_{i+3}s_{i+46}s_{i+64} \\ & + s_{i+3}s_{i+46}b_{i+63} + s_{i+25}s_{i+46}b_{i+63} + s_{i+46}s_{i+64}b_{i+63}. \end{aligned} \tag{3.3}$$

Grain generates a pseudorandom number by xoring $h(s,b)$ and b_i.

(3) Initialization

In the initialization phase, 80-bit key, 64-bit IV, and 16-bit constant are taken as the input to the

internal state.

$$b_i = K_i, \quad 0 \leq i < 80, \tag{3.4}$$

$$s_i = I_i, \quad 0 \leq i < 63, \tag{3.5}$$

$$s_i = 1, \quad 64 \leq i < 80, \tag{3.6}$$

where, K_i and I_i are key byte and IV byte, respectively.

3.3.2 Trivium

Trivium is designed by De Canni‘ere and Preneel. It has been submitted to eSTREAM in 2004 and it has been selected as an eSTREAM Portfolio Profile 2 [26][25]. Trivium has been adopted as an ISO/IEC 29192-2 standard stream cipher. Trivium is a bit-oriented synchronous stream cipher. The key size is 80-bit and the IV size is 80-bit. The size of the internal state is 288-bit. The internal state is represented by $s_1, s_2, \ldots, s_{288}$ from the leftmost bit.

In the initialization phase, key and IV are taken as the input to the internal state. The internal state is updated 1152 times in the initialization phase. In the pseudorandom number generating phase, the internal state is continuously updated and the keystream is generated interleaved with the update of the internal state. The internal state consists of three LFSRs. These LFSRs have different sizes. These sizes are 93-bit, 84-bit, and 115-bit, respectively.

The key, IV, and constants are taken as the input to LFSRs. Figure. 3.2 shows a structure of Trivium. Here, $f(x)$ and $g(x)$ are update functions for LFSRs. The data generated by xoring the outputs of $g(x)$ is not returned in the initialization phase. In the pseudorandom number generating phase, the data generated by xoring the outputs of $g(x)$ is returned as keystream.

Since LFSRs shift one bit per clock, Trivium generates keystream one bit per clock. In similar to Grain, it is easy to speed up by implementing update functions t times parallely and shifting t bits per clock. The number of clock cycles needed for the initialization phase is $1152/t$. The cipher generates t-bit keystream per clock. The elements of the functions are required to be implemented t times. By implementing these functions parallely, it is possible to speed up the cipher without lengthening the critical path. Since 64 bits of the state are not taken as an input to the functions, up to 64 iterations can be computed at a clock.

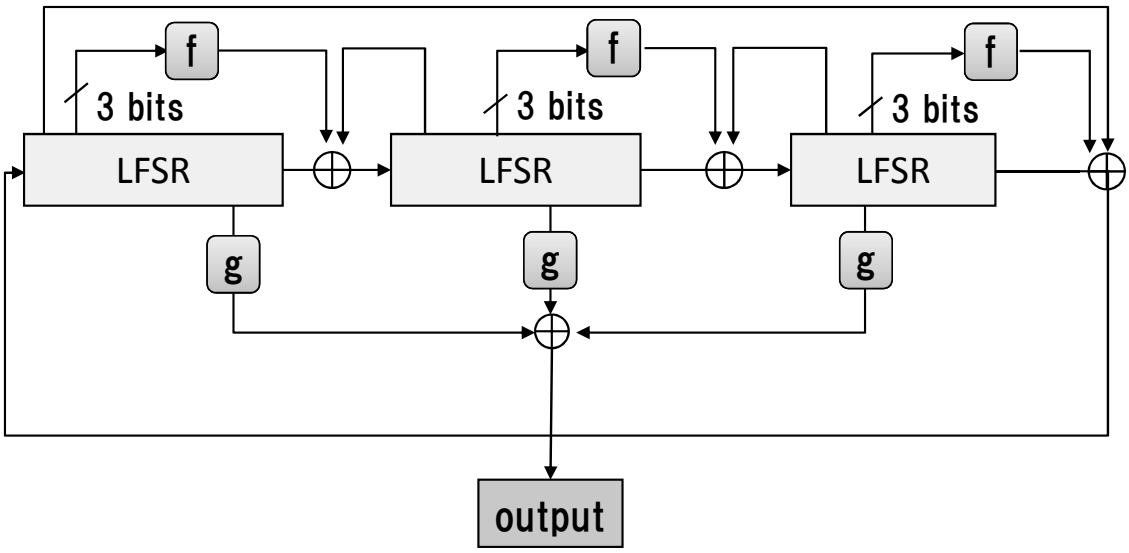


Figure. 3.2 Structure of Trivium

(1) Update function

Algorithm. 3.2 shows the update function. Here, N is the bit length of the desired output.

(2) Output function

Trivium outputs z_i as a pseudorandom bit.

(3) Initialization

In the initialization phase, the internal state is initialized by key, IV, and constant.

$$(s_1, s_2, \ldots, s_{93}) = (K_1, K_2, \ldots, K_{80}, 0, \ldots, 0), \tag{3.7}$$

$$(s_{94}, s_{95}, \ldots, s_{177}) = (IV_1, IV_2, \ldots, IV_{80}, 0, \ldots, 0), \tag{3.8}$$

$$(s_{178}, s_{179}, \ldots, s_{288}) = (0, \ldots, 0, 1, 1, 1). \tag{3.9}$$

Here, K_i and I_i are key byte and IV byte, respectively.

Algorithm. 3.2 Update function of Trivium

Input: the internal state s

Output: the pseudorandom number z_i

1: **for** i = 1, 2, . . . ,N **do**

2: $t_1 = s_{66} + s_{93}$.

3: $t_2 = s_{162} + s_{177}$.

4: $t_3 = s_{243} + s_{288}$.

5: $z_i = t_1 + t_2 + t_3$.

6: $t_1 = t_1 + s_{91} \cdot s_{92} + s_{171}$.

7: $t_2 = t_2 + s_{175} \cdot s_{176} + s_{264}$.

8: $t_3 = t_3 + s_{286} \cdot s_{287} + s_{69}$.

9: $(s_1, s_2, \ldots, s_{93}) = (t_3, s_1, s_2, \ldots, s_{92})$.

10: $(s_{94}, s_{95}, \ldots, s_{177}) = (t_1, s_{94}, s_{95}, \ldots, s_{176})$.

11: $(s_{178}, s_{179}, \ldots, s_{288}) = (t_3, s_{178}, s_{179}, \ldots, s_{287})$.

12: **end for**

3.3.3 Enocoro

Enocoro is designed by Hitachi Ltd. It is adopted as an ISO/IEC 29192-2 standard stream cipher [50]. The key size is 80-bit and the IV size is 64-bit. Enocoro is a byte-oriented synchronous stream cipher. In the initialization phase, key, IV, and constants are taken as the input to the internal state. The size of the internal state is 176-bit. The internal state consists of state and buffer. The buffer is LFSR and its size is 160-bit. The state is represented by a_0, a_1 from the leftmost byte. The buffer is represented by $b_0, b_1, \ldots, b_{19}$ from the leftmost byte. The ith byte of the state and the buffer at time t are represented by a_i^t, b_i^t, respectively.

The internal state is updated 40 times repeatedly. In the pseudorandom number generating phase, the internal state is continuously updated and the keystream is generated interleaved with the update of the internal state. Enocoro outputs 8-bit pseudorandom number per clock. Figure. 3.3 shows a structure of Enocoro.

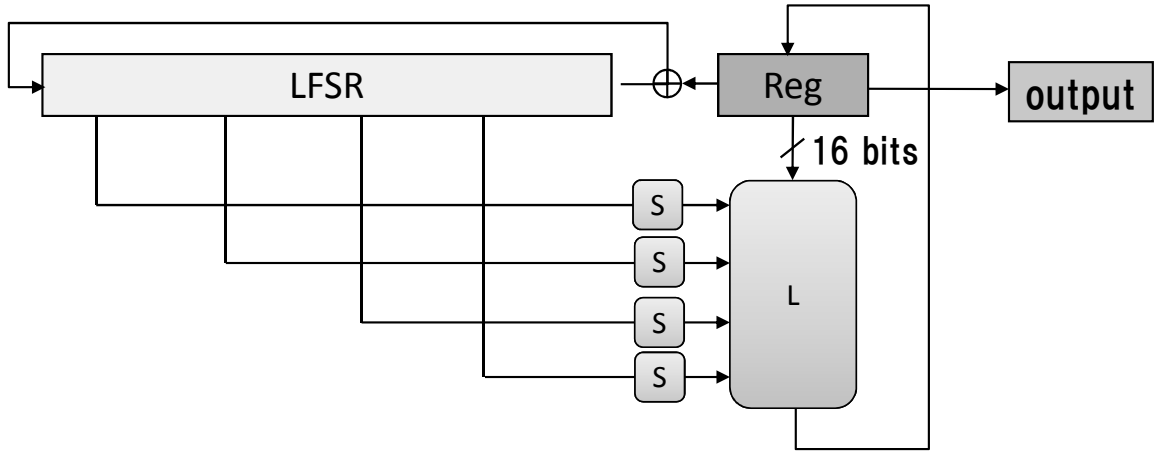


Figure. 3.3 Structure of Enocoro

(1) Update function

The update function of the state is called ρ function. ρ function is defined by the following equations.

$$u_0 = a_0^t \oplus s_8[b_1^t], \tag{3.10}$$

$$u_1 = a_1^t \oplus s_8[b_4^t], \tag{3.11}$$

$$(v_0, v_1) = L(u_0, u_1), \tag{3.12}$$

$$a_0^{t+1} = v_0 \oplus s_8[b_6^t], \tag{3.13}$$

$$a_1^{t+1} = v_1 \oplus s_8[b_{16}^t], \tag{3.14}$$

where, s_8 is a non linear function and maps 8-bit input to 8-bit output. L is a linear function. u_0, u_1, v_0, and v_1 are temporal variables. s_8 is defined by the following equations.

$$y_0 = s_4[s_4[x_0] \oplus 4 \cdot s_4[x_1] \oplus 0\text{x}A], \tag{3.15}$$

$$y_1 = s_4[4 \cdot s_4[x_0] \oplus s_4[x_1] \oplus 0\text{x}5], \tag{3.16}$$

$$s_8[x] = (y_0 || y_1) <<< 1, \tag{3.17}$$

where, x_0 and x_1 represent upper 4-bit of input x and lower 4-bit of input x, respectively. Similarly, y_0 and y_1 represent upper 4-bit of output y and lower 4-bit of output y, respectively. s_4 is a non linear function and maps 4-bit input to 4-bit output. s_4 is defined by the following table.

$$s_4[16] = \{0x1, 0x3, 0x9, 0xA, 0x5, 0xE, 0x7, 0x2, 0xD, 0x0, 0xC, 0xF, 0x4, 0x8, 0x6, 0xB\}. \tag{3.18}$$

Linear function L is defined by a matrix on GF(2^8).

$$\begin{pmatrix} v_0 \\ v_1 \end{pmatrix} = L(u_0, u_1) = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \end{pmatrix}. \tag{3.19}$$

The update function of the buffer is called λ function. λ function is defined by the following equations.

$$b_i^{t+1} = b_{i-1}^t, \quad i \neq 0, 2, 5, 7, \tag{3.20}$$

$$b_0^{t+1} = b_{19}^t \oplus a_0^t, \tag{3.21}$$

$$b_2^{t+1} = b_1^t \oplus b_3^t, \tag{3.22}$$

$$b_5^{t+1} = b_4^t \oplus b_5^t, \tag{3.23}$$

$$b_7^{t+1} = b_6^t \oplus b_{15}^t. \tag{3.24}$$

(2) Output function

Output state a_1 as a pseudorandom number.

(3) Initialization

In the initialization phase, the internal state is initialized by key, IV, and constant.

$$b_i = K_i, \quad 0 \leq i < 10, \tag{3.25}$$

$$b_{i+10} = I_i, \quad 0 \leq i < 8, \tag{3.26}$$

$$b_{i+18} = C_i, \quad 0 \leq i < 2, \tag{3.27}$$

$$a_0 = C_2, \quad a_1 = C_3, \tag{3.28}$$

where, K_i, I_i, and C_i are key byte, IV byte, and constant byte, respectively.

3.3.4 SPONGENT

SPONGENT [19][20] is a lightweight hash function family designed by Bogdanov *et al.* SPONGNET is based on a sponge construction. The sponge construction is a simple iterated construction that takes a variable length input and generates a variable length output. The input message is cut into a fixed length blocks and the block is taken as the input to the internal state. The internal state is updated by a permutation. After that, a part of the internal state is returned as a hash block. The internal state consists of two parts. One is rate r and the other is capacity c.

SPONGENT consists of two phases. The first phase is the absorbing phase. In the absorbing phase, the internal state is initialized with 0. The input message is padded with a bit 1 and then zeros are added until the message length is a multiple of the rate size. Then, the message is cut into blocks that have r bits. The block is xored with the rate interleaved with the application of the permutation. The internal state is taken as the input to the permutation and is updated with the output of the permutation. This phase continues until all padded message blocks are xored with the internal state.

The second phase is the squeezing phase. The rate are returned as a hash block interleaved with the application of the permutation. The internal state is taken as the input to the permutation and is updated with the output of the permutation. The squeezing phase continues until desired bits of hash values are generated.

SPONGENT has five variations that consist of SPONGENT-88, SPONGENT-128, SPONGENT-160, SPONGENT-224, and SPONGENT-256. The bit length of hash values are 88-bit, 128-bit, 160-bit, 224-bit, and 256-bit respectively. SPONGENT-160 is a variation of SPONGENT which has a 176-bit internal state. The internal state consists of a 160-bit capacity and a 16-bit rate. The 16-bit message blocks are xored with the rate in the absorbing phase. Then, the internal state is processed with the permutation π that consists of 90 times iteration of the round function. Figure. 3.4 shows an overview of a structure of SPONGENT-160. Algorithm. 3.3 shows the permutation.

(1) sBoxLayer

In the sBoxLayer, 4-bit input and 4-bit output Sbox S is applied to all bits of the internal state.

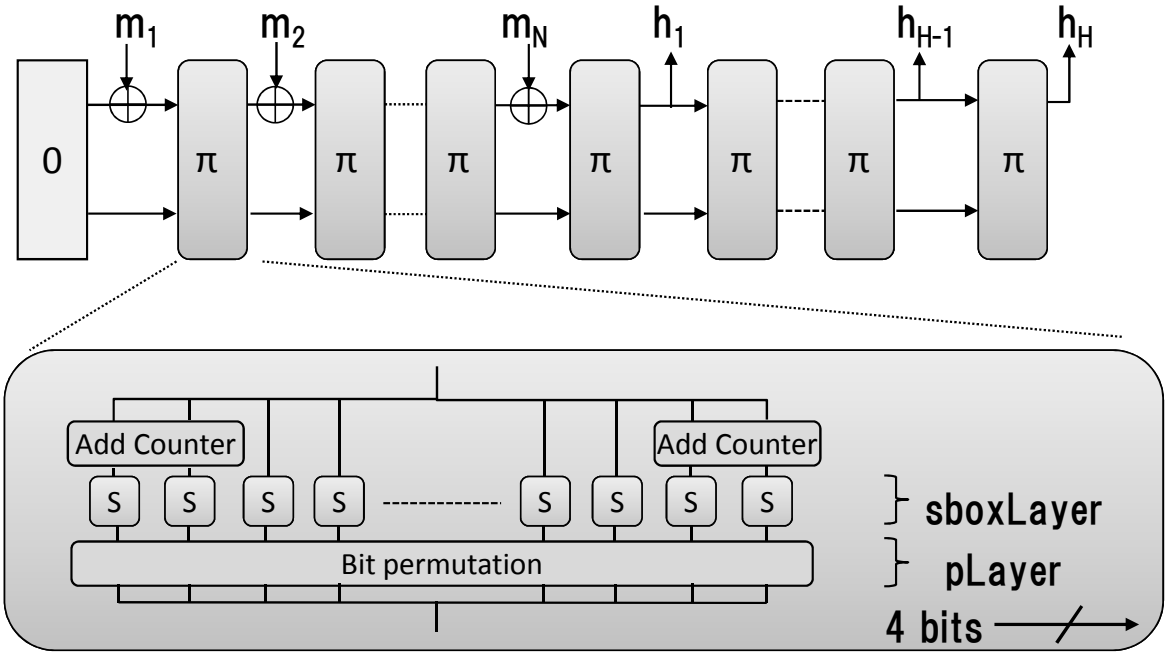


Figure. 3.4 Structure of SPONGENT-160

The Sbox is defined by the following table.

$$S(x) = \{0xE, 0xD, 0xB, 0x0, 0x2, 0x1, 0x4, 0xF, 0x7, 0xA, 0x8, 0x5, 0x9, 0xC, 0x3, 0x6\}. \tag{3.29}$$

(2) pLayer

In the pLayer, bit position j of the internal state is moved to bit position $P(j)$.

$$P(j) = j \times b/4 \mod 175, \quad \text{if} \quad j \in \{0, 1, 2, \ldots, 174\}, \tag{3.30}$$

$$P(j) = 175, \quad \text{if} \quad j = 175. \tag{3.31}$$

(3) lCounter

lCounter is a round constant and is generated by 7-bit LFSR. The LFSR is defined by the primitive

Algorithm. 3.3 Permutation of SPONGENT-160

Input: the internal state STATE, the round constant lCounter

Output: the updated internal state

1: **for** i = 1, 2, . . . ,90 **do**

2: STATE = Reverse_ lCounter (i) $\oplus$ STATE $\oplus$ lCounter (i).

3: STATE = sBoxLayer (STATE).

4: STATE = pLayer (STATE).

5: **end for**

polynomial x^7+x^6+1 and the initial value is $0x1000101$. Reverse_lCounter is the value of lCounter with reversed bit order.

3.3.5 PHOTON

PHOTON is a family of lightweight hash functions designed by Guo *et al.* PHOTON is based on a sponge construction. The internal state consists of two parts. One is rate r and the other is capacity c .

PHOTON consists of two phases. The first phase is the absorbing phase. In the absorbing phase, the internal state is initialized with 0. The input message is padded with a bit 1 and then zeros are added until the message length is a multiple of the rate size r. Then, the message is cut into blocks that have r bits. The block is xored with the rate interleaved with the application of the permutation. The internal state is taken as the input to the permutation and is updated with the output of the permutation. This phase continues until all padded message blocks are xored with the internal state.

The second phase is the squeezing phase. The rate is returned as a hash block interleaved with the application of the permutation. The internal state is taken as the input to the permutation and is updated with the output of the permutation. The squeezing phase continues until desired bits of hash values are generated.

PHOTON has five variations that consist of PHOTON-88/20/16, PHOTON-128/16/16, PHOTON-160/36/36, PHOTON-224/32/32, and PHOTON-256/32/32. The bit length of hash

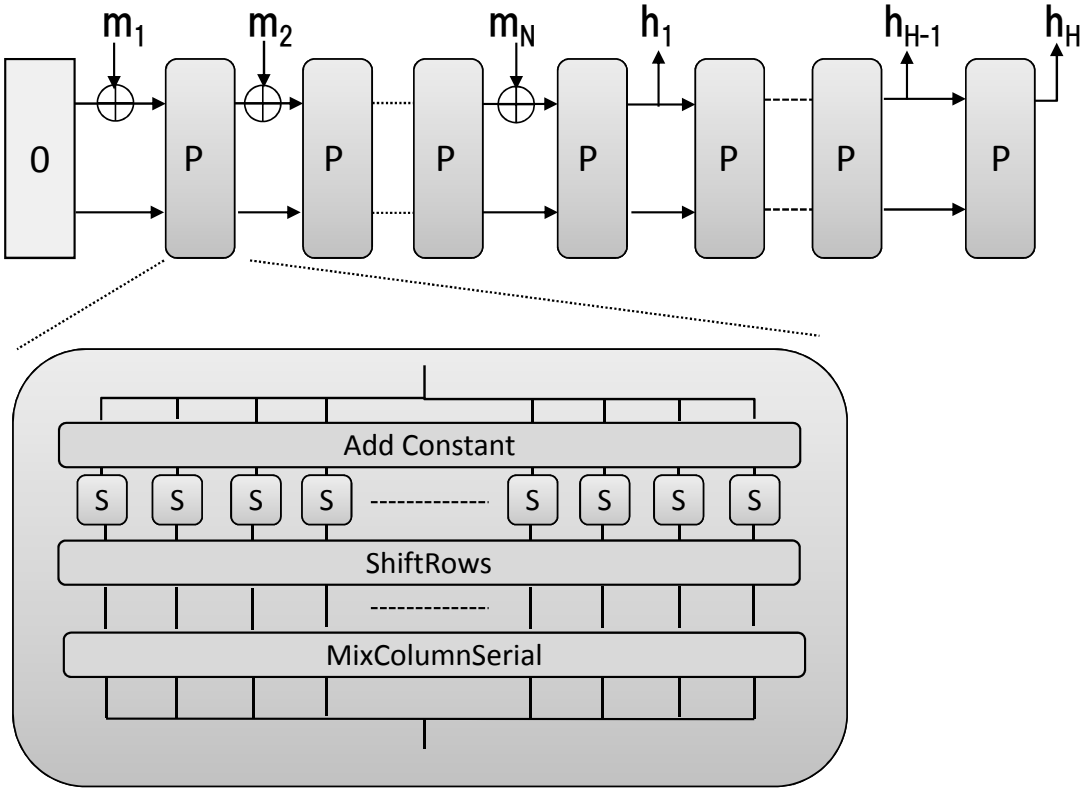


Figure. 3.5 Structure of PHOTON-160/36/36

values are 88-bit, 128-bit, 160-bit, 224-bit, and 256-bit, respectively. PHOTON-160/36/36 is a variation of PHOTON which has a 196-bit internal state. The internal state consists of a 160-bit capacity and a 36-bit rate. The 36-bit message blocks are xored with the rate in the absorbing phase. Then, the internal state is processed with the permutation P that consists of 12 times iteration of the round function. Figure. 3.5 shows a structure of PHOTON-160/36/36. Algorithm. 3.4 shows the permutation. The state is treated as $d \times d$ matrix S that has s-bit cells.

(1) AddConstant

At vth round, a round constant $RC(v)$ are xored with the internal state $S[i, 0]$. $RC(v)$ is defined by

Algorithm. 3.4 Permutation of PHOTON-160/36/36

Input: the internal state S, the round constant RC

Output: the updated internal state S

1: **for** i = 1, 2, . . . ,12 **do**
2: S = AddConstant(S, RC).
3: S = SubCells(S).
4: S = ShiftRows(S).
5: S = MixColumnsSerial(S).
6: **end for**

the following table.

$$RC(v) = [0x1, 0x3, 0x4, 0xE, 0xD, 0xB, 0x6, 0xC, 0x9, 0x2, 0x5, 0xA]. \tag{3.32}$$

Then, another constant $IC_d(i)$ is xored with the internal state $S[i, 0]$. $IC_d(i)$ is defined by the following table.

$$IC_d = [0x0, 0x1, 0x2, 0x5, 0x3, 0x6, 0x4]. \tag{3.33}$$

AddConstant layer is summarized with the following equation.

$$S'[i, 0] = S[i, 0] \oplus RC(v) \oplus IC_d(i), \quad 0 \leq i < 7. \tag{3.34}$$

(2) SubCells

4-bit input and 4-bit output Sbox is applied to the internal state.

$$S'[i, j] = Sbox(S[i, j]), \quad 0 \leq i, j < 7. \tag{3.35}$$

Sbox is defined by the following table.

$$Sbox(16) = \{0xC, 0x5, 0x6, 0xB, 0x9, 0x0, 0xA, 0xD, 0x3, 0xE, 0xF, 0x8, 0x4, 0x7, 0x1, 0x2\}. \tag{3.36}$$

(3) ShiftRows

Each row of the internal state rotates to the right. ShiftRows is defined by the following equation.

$$S'[i, j] = S[i, (j + i) \bmod d], \quad 0 \leq i, j < 7. \tag{3.37}$$

(4) MixColumnsSerial

Each column of the internal state is updated by applying the matrix A for 7 times. MixColumnsSerial is defined by the following equation.

$$(S'[0, j], \ldots, S'[6, j])^T = A^7 \times \ (S[0, j], \ldots, S[6, j])^T, \quad 0 \leq j < 7, \tag{3.38}$$

where, T means transposition. Matrix A is defined by the following equation.

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 4 & 6 & 1 & 1 & 6 & 4 \end{pmatrix}. \tag{3.39}$$

The irreducible polynomial is $x^4 + x + 1$.

3.3.6 QUARK

QUARK [3][6] is a lightweight hash function family designed by Aumasson *et al.* QUARK is based on a sponge construction. The internal state consists of two parts. One is rate r and the other is capacity c .

QUARK consists of two phases. The first phase is the absorbing phase. In the absorbing phase, the state is initialized with 0. The input message is padded with a bit 1 and then zeros are added until the message length is a multiple of the rate size r. Then, the message is cut into blocks that have r bits. The block is xored with the rate interleaved with the application of the permutation. The internal state is taken as the input to the permutation and is updated with the output of the

permutation. This phase continues until all padded message blocks are xored with the internal state.

The second phase is the squeezing phase. The rate are returned as a hash block interleaved with the application of the permutation. The internal state is taken as the input to the permutation and is updated with the output of the permutation. The squeezing phase continues until desired bits of hash values are generated.

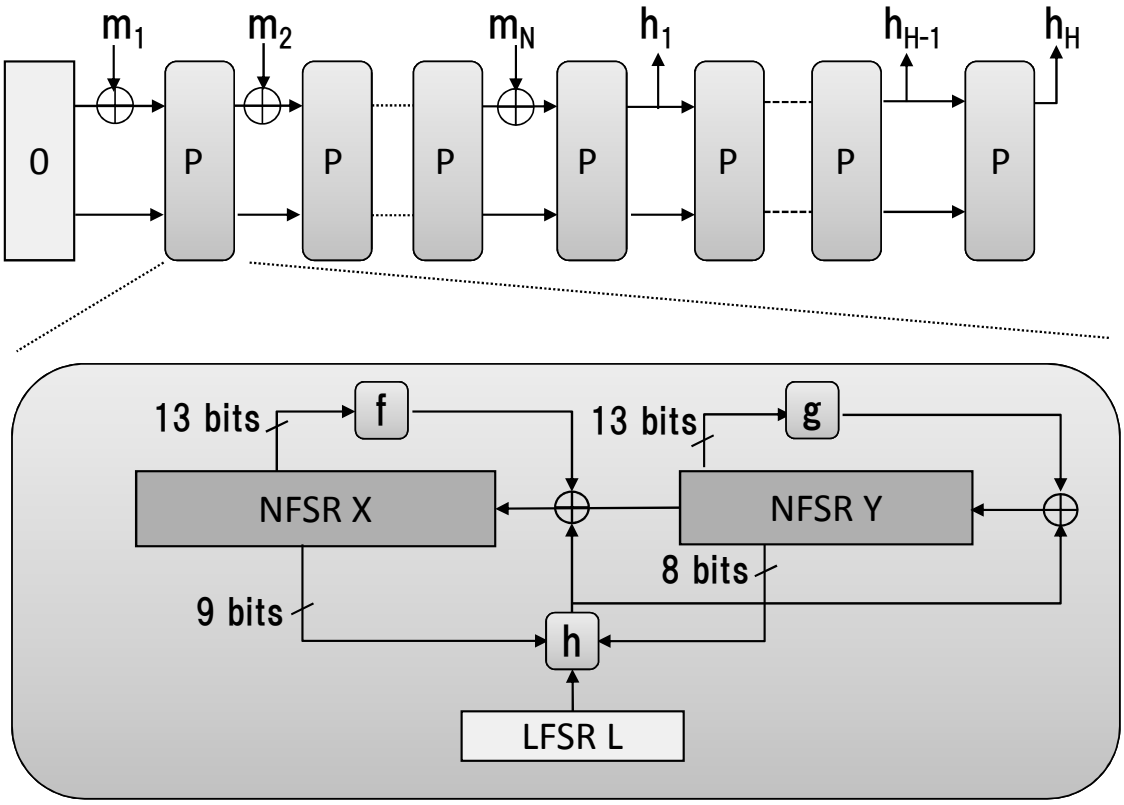


Figure. 3.6 Structure of D-QUARK

QUARK has three variations that consist of U-QUARK, D-QUARK, and S-QUARK. The bit length of hash values are 128-bit, 160-bit, and 224-bit, respectively. D-QUARK is a variation of QUARK which has a 176-bit internal state. The internal state consists of a 160-bit capacity and a

Algorithm. 3.5 Round function of D-QUARK

Input: the LFSR L, the NFSRs X and Y

Output: the updated LFSR L, the NFSRs X and Y

1: **for** i = 1, 2, . . . ,704 **do**

2: $h^i = h(X, Y, L)$.

3: $(X'_0, X'_1, \ldots, X'_{79}) = (X_1, X_2, \ldots, X_{79}, Y_0 + f(X) + h^i)$.

4: $(Y'_0, Y'_1, \ldots, Y'_{79}) = (Y_1, Y_2, \ldots, Y_{79}, g(Y) + h^i)$.

5: $(L'_0, L'_1, \ldots, L'_9) = (L_1, L_2, \ldots, L_9, p(L))$.

6: **end for**

16-bit rate. The 16-bit message blocks are xored with the rate in the absorbing phase. Then, the internal state is processed with the permutation P that consists of 704 times iteration of the round function. Figure. 3.6 shows an overview of a structure of D-QUARK.

The NFSRs are represented by $X = (X_0, X_1, \ldots, X_{79})$ and $Y = (Y_0, Y_1, \ldots, Y_{79})$. The bit length of NFSRs are 80-bit. The LFSR is represented by $L = (L_0, L_1, \ldots, L_9)$. The bit length of LFSR is 10-bit.

(1) Round function

The round function consists of Non-linear Boolian functions f, g, h, and a linear Boolian function p. Algorithm. 3.5 shows the round function.

The output of the function f is defined by the following term.

$$\begin{aligned} f(X) = {} & X_0 + X_{11} + X_{18} + X_{27} + X_{36} + X_{42} + X_{47} + X_{58} + X_{64} + X_{67} + X_{71} + X_{71}X_{79} + \\ & X_{42}X_{47} + X_{11}X_{19} + X_{58}X_{67}X_{71} + X_{27}X_{36}X_{42} + X_{11}X_{36}X_{58}X_{79} + X_{42}X_{47}X_{67}X_{71} + \\ & X_{19}X_{27}X_{71}X_{79} + X_{47}X_{58}X_{67}X_{71}X_{79} + X_{11}X_{19}X_{27}X_{36}X_{42} + X_{27}X_{36}X_{42}X_{47}X_{58}X_{67}. \end{aligned} \tag{3.40}$$

The output of the function g is defined by the following term.

$$\begin{aligned} g(Y) = {} & Y_0 + Y_9 + Y_{20} + Y_{25} + Y_{38} + Y_{44} + Y_{47} + Y_{69} + Y_{63} + Y_{67} + Y_{69} + \\ & Y_{69}Y_{78} + Y_{44}Y_{47} + Y_9Y_{19} + Y_{54}Y_{67}Y_{69} + Y_{25}Y_{38}Y_{44} + Y_9Y_{38}Y_{54}Y_{78} + Y_{44}Y_{47}Y_{67}Y_{69} + \\ & Y_{19}Y_{25}Y_{69}Y_{78} + Y_{47}Y_{54}Y_{67}Y_{69}Y_{78} + Y_9Y_{19}Y_{25}Y_{38}Y_{44} + Y_{25}Y_{38}Y_{44}Y_{47}Y_{54}Y_{67}. \end{aligned} \tag{3.41}$$

The output of the function h is defined by the following term.

$$\begin{aligned} h(X, Y, L) = {} & L_0 + X_1 + Y_2 + X_5 + Y_{12} + Y_{24} + X_{35} + X_{40} + X_{48} + Y_{55} + Y_{61} + \\ & X_{72} + Y_{79} + Y_4X_{68} + X_{57}X_{68} + X_{68}Y_{79} + Y_4X_{35}X_{57} + \\ & Y_4X_{57}X_{68} + Y_4X_{57}Y_{79} + L_0X_{35}X_{57}Y_{79} + L_0X_{35}. \end{aligned} \tag{3.42}$$

The output of the function p is defined by the following term.

$$p(L) = L_0 + L_3. \tag{3.43}$$

3.4 Implementation Platform

3.4.1 FPGA

FPGA is a configurable integrated circuit. FPGA consists of an array of programmable logic blocks, reconfigurable interconnections, and I/O pad. The logic block is configured to perform combinational functions. The logic block consists of several logical cells and memory elements, such as FFs, and blocks of memory. The logical cell is called Slice or ALM. The slice consists of two or four cells. The cell mainly consists of 4-input LUT and a FF. In these days, some FPGA uses 6-input LUTs.

The behavior of the FPGA is defined by a designer using HDL, such as Verilog-HDL, and VHDL. A netlist is generated by a design tool. The netlist is fitted to the FPGA by a process called place and route. The process is done by a place and route tool. After the process, a binary file is generated. The file defines the behavior of the FPGA and is transferred to the FPGA. Reconfigurations are done by changing the functionalities of the cells and the interconnections of the blocks.

3.4.2 ASIC

ASIC is an integrated circuit that is customized for a single behavior. The development process of ASIC is similar to that of FPGA. A designer describes a functionality of ASIC using HDL to define the behavior of ASIC. This design is called RTL design. Then, a designer verifies functionality of the RTL design using a software tool. This process is called logic simulation. After that, the RTL design is transformed into a lower level construction by logic synthesis using standard cell library that describes characteristics of basic gates, such as 2-input AND, 2-input OR, and invertor. The characteristics of the standard cells are different according to the manufactures of ASIC. After the synthesis, a gate level netlist is generated. The gate level netlist contains a set of the standard cells and electrical connections between the standard cells. The gate level netlist is taken as the input to a placement and route tool. The tool tries to find a placement of the standard cells and physical connections of the standard cells taking into consideration of several constrains in terms of area and time. Finally, the photomask of the circuit is released for fabrication.

3.5 Comparison Metrics for Hardware Performance

To compare hardware performance of the circuits using a single value, previous studies have used a product of different kinds of metrics, such as area requirements, power consumption, energy, the number of clock cycles, and the operating frequency. However, the important metric is different according to the devices. For example, in the case of an active tag, the energy is an important metric. Since the tag has a battery, the life of the battery is affected by the energy. In the case of a passive tag, the power consumption is an important metric. The tag does not operate when the power consumption of the tag exceeds the received power from a reader.

We deduce that a triple of parameters is suitable for performance evaluation for a passive tag. A triple consists of area requirements, power consumption, and the number of clock cycles. The power consumption of the circuit is given by μW. The power consumption affects to the maximum read distance of the tag. The area requirements of the circuit are given by GE. The area requirements affects to the cost of the tag. The number of clock cycles does not depend on the clock

frequency. The number of clock cycles affects to the operating time of the tag. This triple allows calculation of a product metric such as the energy.

3.5.1 Area Requirements

Area requirements indicate an amount of silicon needed to implement the design of a circuit. From an economic point of view, the amount of silicon is an important issue since the production cost increases in accordance with the size of the integrated circuits. The area requirements of the circuit are typically measured by μm^2. However, this metric depends on a fablication process of the circuit. Since GE is independent from the fablication process, this is a more usable metric. GE is calculated by dividing the area requirements by the area occupied with a two-input NAND gate.

3.5.2 Operating Time

There are mainly two metrics to measure time related requirement. One is throughput and the other is clock cycles. Throughput is measured in bit per second manner and is calculated by dividing the product of the bit size of the output and the operating frequency by the number of clock cycles. Usually, the throughput is calculated under the condition that the circuit operates at the maximum frequency. The circuit achieves high throughput as the operating frequency is high. However, the operating frequency is low in the case of tags. Therefore, the throughput is not a suitable measure for the tags.

The second one is the number of clock cycles needed to execute cryptographic process including data loading and storing. This is a simple measure and does not depend on another metric such as the operating frequency. The operating frequency of the tags differs from the kind of tags or the application of the tag. Therefore, the number of clock cycles is a suitable measure for the tag.

3.5.3 Power Consumption

The operating power is supplied to a passive tag by a reader over the air interface. Weak electro magnetic wave is emitted by the reader's antenna and is received by the tag's antenna. The tag's

antenna induces a voltage from the electromagnetic wave. This voltage depends on the distance between the reader and the tag. In the case of the RFID system operating in UHF band, the power received by the tag is inversely proportional to the square of the distance between the tag and the reader. The maximum read distance of the tag is affected by the power consumption of the tag [39].

$$P_{tag} = \eta \cdot P_{reader} \cdot G_t \cdot G_r / (4\pi d/\lambda)^2, \tag{3.44}$$

where, P_{tag} is the power received by the tag. P_{reader} is the power transmitted by the reader. η is P_{reader}-to-P_{tag} power conversion efficiency. η reflects the power reception efficiency of microwaves or the rectification conversion efficiency. G_t and G_r are the free space gain of the reader's antenna and tag's antenna, respectively. λ is the carrier frequency wavelength. d is the distance between the tag and the reader. The maximum read distance of the tag affects the application of the tag.

Part III

Selection and Implementation of Cryptographic Components

Chapter 4

Hardware Architecture of Cryptographic Algorithm for RFID Tag

4.1 Introduction

To protect the tag from the security and the privacy risks, tag authentication protocols based on cryptographic algorithms have been studied [51]. To the best of our knowledge, hash functions and pseudorandom number generators are widely used as the cryptographic algorithm.

We deduce that the important metrics for the tag are cost, maximum read distance, and operating time. These physical metrics are directly affected by hardware performance of a cryptographic circuit. The cost is affected by the area requirements of the circuit. The maximum read distance is affected by the power consumption of the circuit. The operating time is affected by the number of clock cycles needed to complete the cryptographic process in the circuit.

As a previous result, Saarinen *et al.* have studied some requirements for the cryptographic algorithms for the tag from hardware performance of the cryptographic circuit [96]. They have picked up EPCglobal Class-1 Generation-2 [34] and have studied that some lightweight hash functions. They have mentioned that the lightweight hash functions they have studied were inappropriate for

a tag and the lightweight stream ciphers might be useful for a tag. However, they have not taken care of the security strength. They have evaluated the number of clock cycles needed to generate 16-bit output. We deduce that this bit length is too small to be applied in a real world as a response of a tag collides with another response of another tag with a probability of $1/2^8$.

The hardware performance are largely affected by hardware architecture of the cryptographic circuit. Thus, we study hardware architecture, which is suitable for the tag, of the cryptographic algorithms. We focus on lightweight hash functions and lightweight stream ciphers that are widely used for a cryptographic algorithm for tag authentication.

4.2 Basic Hardware Architectures

There are three major hardware architectures. Namely, the serialized architecture, the parallel architecture, and the sequential architecture.

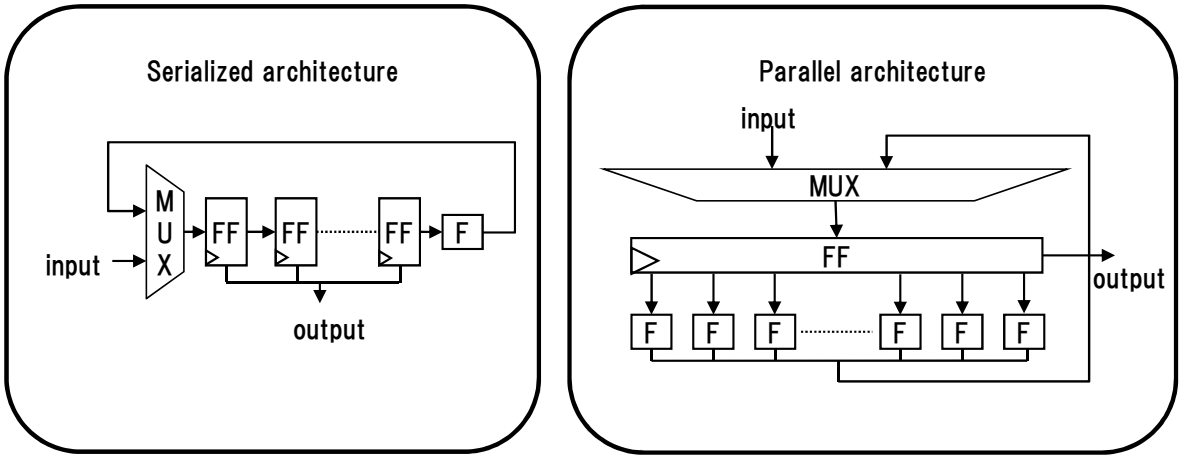


Figure. 4.1 Serialized architecture and parallel architecture

4.2.1 Parallel Architecture

Figure. 4.1 shows an overview of the parallel architecture. In Figure. 4.1, FF, F, and MUX mean a register, a function, and a multiplexer, respectively. The parallel architecture is characterized

by the property that all bits of data stored in registers are updated by the functions at the same time. The parallel architecture performs one round function of the cryptographic process within one clock cycle. It is also possible to perform several round functions within one clock cycle by implementing the round function several times. The parallel architecture achieves small number of clock cycles while the area requirements of the circuit are slightly large.

4.2.2 Serialized Architecture

Figure. 4.1 shows an overview of the serialized architecture. The serialized architecture is characterized by the property that data bits are stored in register chain and are updated by the function in a bit or byte wise fashion. The serialized architecture performs a fraction of the round function within one clock cycle. Namely, it takes several number of clock cycles to process one round function. Since the control circuit will be more complex, the area occupied with the control circuit will increase. While this architecture achieves low area requirements of the circuit, the number of clock cycles are large.

4.2.3 Sequential Architecture

4.2.3.1 Power Reduction Method

One of the important physical metrics for the tag is the maximum read distance. The maximum read distance is affected by the power consumption of the tag. Generally speaking, there is a linear relationship between the power consumption of the circuit and the number of signal toggle [28].

$$P_{total} = P_{toggle} \cdot C_L \cdot V_{dd}^2 \cdot f + I_{leak} \cdot V_{dd}, \tag{4.1}$$

where, P_{toggle}, C_L, V_{dd}, f, and I_{leak} are number of the signal toggles, loading capacity, power supply voltage, clock frequency, and a leakage current, respectively. Therefore, reduction of the number of the signal toggles leads power reduction of the circuit.

4.2.3.2 Sequential Architecture

As the cryptographic circuit is connected to another circuit in the tag, a buffer will be used to store data that are taken as the input to the cryptographic circuit and are returned as output from the

cryptographic circuit temporaly. Usually the buffer stores data only when the data are transferred between the cryptographic circuit and the outer circuit.

As the buffer is not used when the internal state is updated with the output of the update function, the buffer can be used as a temporal register to store a part of the internal state. By storing a part of the internal state in the buffer, the internal state can be updated with the output of the update function sequentially. We deduce that a hardware architecture updating the internal state sequentially can achieve low number of the signal toggles.

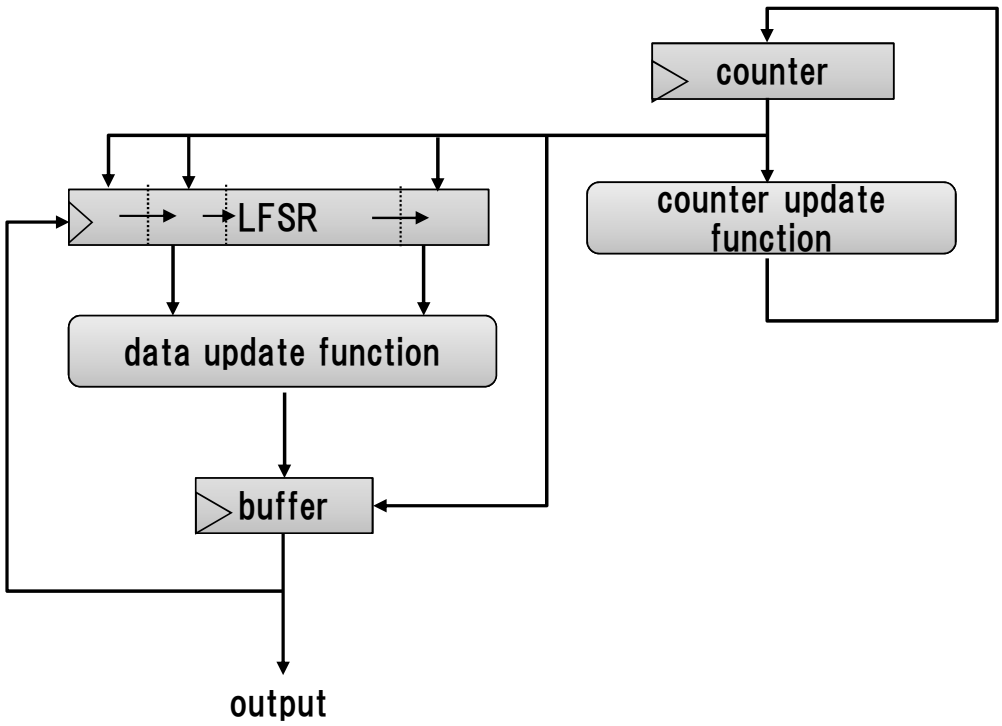


Figure. 4.2 Sequential Architecture

Figure. 4.2 shows an overview of the sequential architecture. In the figure, the counter is a register to store counter value. The counter update function is the update function of the counter value. The output of the counter is taken as the control signal to the LFSR and the buffer.

We describe update steps of the internal state. First, the output of the update function is taken as the input to the buffer. Second, an M-bit block of LFSR is shifted to the right. Here, M is the size of the buffer. This step is repeated until all bits of LFSR shift to the right. Finally, data stored

in the buffer are taken as the input to the leftmost M-bit of LFSR.

While this architecture is serialized-based architecture, the sequential architecture is different from the serialized architecture in the following sense. While all bits of data stored in FFs in the serialized architecture shift simultaneously, data stored in FFs in the sequential architecture shift sequentially.

Design of control circuit

As shown in Figure. 4.2, LFSR and the buffer are controlled by the counter value. Generally speaking, the counter value is incremented every clock. The counter is initialized with 0 and incremented by 1. The signal line of the counter has usually multiple bit width while it depends on the number of processing steps. Therefore, there is a concern about glitch on the counter. Glitch is an unnecessary transition of the signal and it causes increasing of the power consumption.

We have designed and implemented a simple counter described above. Then, we have simulated the behavior of the signal toggle of the circuit. We have designed the counter in Verilog-HDL. We have used Synopsys Design Compiler (Version C-2009.06-SP5) and TSMC 90 nm high performance library to generate a netlist. We have used Simvision version 05.83-s015 to simulate the netlist.

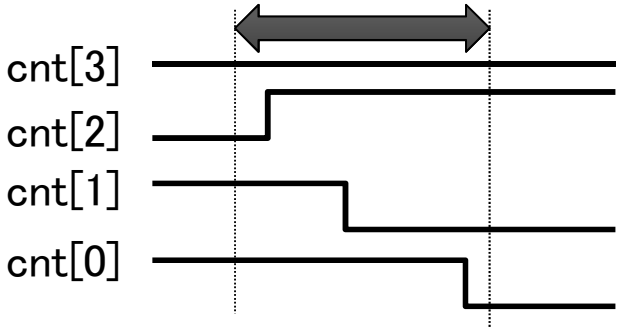


Figure. 4.3 Glithces on the signal line of the counter

Figure. 4.3 shows detail behavior of the signal line of the counter switching from 11 to 12. The bit width of the signal line are 4 and these lines are indicated by cnt[3], cnt[2], cnt[1], and cnt[0] from high order bit. The signal level of the signal lines changes one by one with time delay.

Therefore, the counter value does not change from 11 to 12 directly. The counter value changes through transitional values, such as 15, and 13. From these results, it is clear that glitches occur in the signal line of the counter.

If a glitch occurs in the signal line of the counter, glitches occure in the datapaths. This is because that datapaths in the sequential architecture are controlled by the counter value. Since it is known that power consumption caused by glitches are large, the power consumption of the cryptographic circuit will be large [70].

Thus, we design a counter that does not cause glitches on the signal line of the counter. As shown in Figure. 4.3, we know that the signal level of the counters changes one by one. This means that the counter value changes through several steps and the Hamming distance between two continuous steps is 1. Therefore, we design a cyclic counter. This counter has a property that the Hamming Distance between two continuous values are always 1. As this counter does not have opportunity to take temporal values, we can reduce glitch occurrences on the signal lines of the counter.

We show an example of the counter on Table 4.1. The bit width is 4 and the number of steps are 14. If the number of steps equal to 2^n (n is the bit width of the counter), the Gray code can be used. Gray code is a binary code and two continuous values differ only 1 bit [42]. This property is similar to the characteristic that our counter has.

4.3 Hardware Architecture of Lightweight Cryptographic Algorithms

The cryptographic circuit consists of FFs and combinational logics. FFs are used to store the internal data. The combinational logics are used for transform functions. When we focus on the lightweight cryptographic algorithms, a large part of the area requirements of the cryptographic circuits are occupied with FFs and these areas are irreducible. Most of the lightweight cryptographic algorithms adopt transform functions that can be implemented with a few combinational logics. On the other hand, FFs are area-consuming components. Therefore, area requirements of the lightweight cryptography will not drastically be different according to the hardware architec-

Table 4.1 Counter with constant Hamming Distance

Round	Counter value
#	bit
0	0000
1	0001
2	0011
3	0111
4	0101
5	1101
6	1001
7	1011
8	1010
9	1000
10	1100
11	0100
12	0110
13	0010

ture.

On the other hand, to ensure the security of the lightweight cryptographic algorithms, large round number, which defines the updating times of the internal state with the update function, is likely to be adopted. When the parallel architecture is used to implement a lightweight cryptographic algorithm, one clock cycle is needed to update all bits of the internal state one time. When the serialized architecture is used to implement the lightweight cryptographic algorithm, more number of clock cycles are required to update all bits of the internal state one time since only a fraction of the round function are implemented. Furthermore, when the sequential architecture is used to implement the lightweight cryptographic algorithm, much more number of clock cycles will be required to update all bits of the internal state one time since the internal state are

updated sequentially. Therefore, the number of clock cycles largely depends on the hardware architecture.

Thus, we study hardware architecture that is suitable for implementation of the lightweight cryptographic algorithms on a tag based on hardware performance in the sense of the number of clock cycles. The number of clock cycles can be evaluated without implementing the lightweight cryptographic algorithms.

4.3.1 Hardware Architecture for Lightweight Hash Functions

Table 4.2 Hardware evaluation results of the lightweight hash functions

	Algorithms	Architecture	Clock cycles
Hash functions	SPONGENT-160	Parallel	1534
	SPONGENT-160 [21]	Serialized	75240
	SPONGENT-160 [21]	Sequential	902880
	D-QUARK	Parallel	1500
	D-QUARK [3]	Serialized	13380
	D-QUARK [3]	Sequential	160516
	PHOTON-160/36/36	Parallel	106
	PHOTON-160/36/36 [48]	Serialized	11998
	PHOTON-160/36/36 [48]	Sequential	167842
	Keccak	Parallel	64
	Keccak [62]	Serialized	1240
	Keccak [62]	Sequential	121240

Table 4.2 shows evaluation results of the lightweight hash functions in the sense of the number of clock cycles. Since the number of clock cycles are affected by bit length of the message and the hash value, we evaluate the number of clock cycles needed to input a 160-bit message and output a 160-bit hash value.

The serialized architecture of the lightweight hash functions requires a large number of clock cycles [76]. Since the lightweight hash functions take a short message block as the input and returns a short hash block as the output, the numbers of the message block and the hash block are large. The internal state is updated by the round function for defined times interleaved with

the message block input and the hash block output. Since only a few components of the update functions are implemented, the circuit processes a few bits of the internal state per clock. When the lightweight hash functions are implemented with the sequential architecture, the internal state shifts sequentially, and then a few bits of the internal state are updated with the update function. Therefore, the lightweight hash functions implemented with the serialized architecture and the sequential architecture take a large number of clock cycles.

From our evaluation results, 902880 clock cycles are required to generate 160-bit hash value with SPONGENT-160 implemented with the sequential architecture. As an IC designed for a tag in a market operates with 125 kHz clock signal [9], it takes about 9 seconds to finish the cryptographic process.

4.3.2 Hardware Architecture for Lightweight Stream Ciphers

Table 4.3 Hardware evaluation results of the lightweight stream ciphers

	Algorithms	Architecture	Clock cycles
Stream ciphers	Grain-80	Parallel	82
		Sequential	299
	Trivium	Parallel	167
		Sequential	1595
	Enocoro-80	Parallel	90
		Sequential	418

Table 4.3 shows evaluation results of the lightweight stream ciphers in the sense of the number of clock cycles. Since the number of clock cycles are affected by bit length of pseudorandom number, we evaluate the number of clock cycles needed to output 160-bit pseudorandom number.

As the lightweight stream ciphers are LFSR-based algorithms, the parallel architecture and the serialized architecture for the lightweight stream cipher are the same. When the lightweight stream ciphers are implemented with the sequential architecture, the internal state shifts sequentially, and then a few bits of the internal state are updated with the update function. However, the lightweight stream ciphers still achieve high speed performance. The lightweight stream ciphers consist of the initialization process, which does not generate output, and the pseudorandom number generation

process that generates output every round. Therefore, the initialization process takes dominant part of the number of clock cycles. On the other hand, as the algorithms generate pseudorandom bits every round after the initialization process, these algorithms achieve high speed performance.

4.4 Conclusion

Hardware performance of the cryptographic algorithms in the sense of operating time depends on hardware architecture. This operating time will restrict applications of the tag. Therefore, we have studied hardware architecture of the cryptographic algorithm for a tag from a viewpoint of the operating time. We focus on the lightweight hash functions and the lightweight stream ciphers that are widely used for tag authentication protocols. We have picked up the parallel architecture, the serialized architecture, and the sequential architecture.

From our evaluation results, it has been cleared that the number of clock cycles differ 600 times according to the hardware architecture. Our results show that the parallel architecture seems suitable for the lightweight hash functions. The parallel architecture and the sequential architecture seems suitable for the lightweight stream ciphers.

Chapter 5

Selection of Cryptographic Algorithm

Publication data

Shugo Mikami, Dai Watanabe, and Kazuo Sakiyama, "A Performance Evaluation of Cryptographic Algorithms on FPGA and ASIC on RFID Design Flow," In Proc. International Conference on Information and Communication Technology (ICoICT'16), IEEE, 2016.

5.1 Introduction

Hardware performance of the lightweight cryptographic algorithms in the sense of the number of clock cycles largely depends on the hardware architecture of the lightweight cryptographic algorithms. Therefore, in Chapter 4, we study hardware architectures of the lightweight cryptographic algorithms. A hardware performance of the lightweight cryptographic algorithms in the sense of the number of clock cycles needed to complete the cryptographic process can be evaluated without implementation of the lightweight cryptographic algorithms. On the other hand, hardware performance of the lightweight cryptographic algorithms in the sense of area requirements and the number of signal toggles, which is a main cause of power consumption, can be evaluated after implementing these algorithms.

The hardware performance of the lightweight cryptographic algorithms depend not only the algorithms but also evaluation tools, such as platform, compiler, and standard cell library. As

previous studies, the lightweight block ciphers have been targeted and comparative studies of them in the sense of hardware performance have been studied. In the most recent work of Kerckhoff *et al.*, area, power consumption, throughput, and energy of 6 block ciphers have been evaluated [58]. Batina *et al.* have reported area, power consumption, and energy of 11 block ciphers [16]. These studies focus on a fair comparison of AES [80] to recently developed lightweight block ciphers. Hardware performance of some cryptographic algorithms achieving different security strengths has been discussed in [41].

However, these studies have not taken care of the interface of the cryptographic circuit. The data width of the interface has set arbitrarily. As the cryptographic circuit in the tag is connected to another circuit to realize the authentication process, we deduce that it is important to design the interface considering the system that uses the circuit.

In addition, when we focus on the lightweight hash functions and the lightweight stream ciphers that are widely used as a cryptographic algorithm on the tag authentication protocols, a comparative analysis of these algorithms based on hardware performance has not been studied. Thus, to select the cryptographic algorithms that are suitable for a tag, we implement and evaluate hardware performance of the lightweight hash functions and the lightweight stream ciphers.

5.2 Design of Interface for Cryptographic Circuit

5.2.1 Overview of RFID Tag

Figure. 5.1 shows an overview of a typical RFID tag. To simplify the figure, only modules in the data path are shown. The analog part performs the demodulation of the received data and extracts the power from the RF field mainly. The analog power block extracts the energy from the RF field and supplies the power for the blocks. The analog clock block generates clock signals needed for the digital processing. The digital part performs the encoding/decoding of the data format and cryptographic process mainly.

Since the input data and the output data of the cryptographic module are used in the other module, these data are stored in SRAM and are accessed from several modules. To take low-area requirements and achieve low-power consumption of the tag, we have selected SRAM parameter-

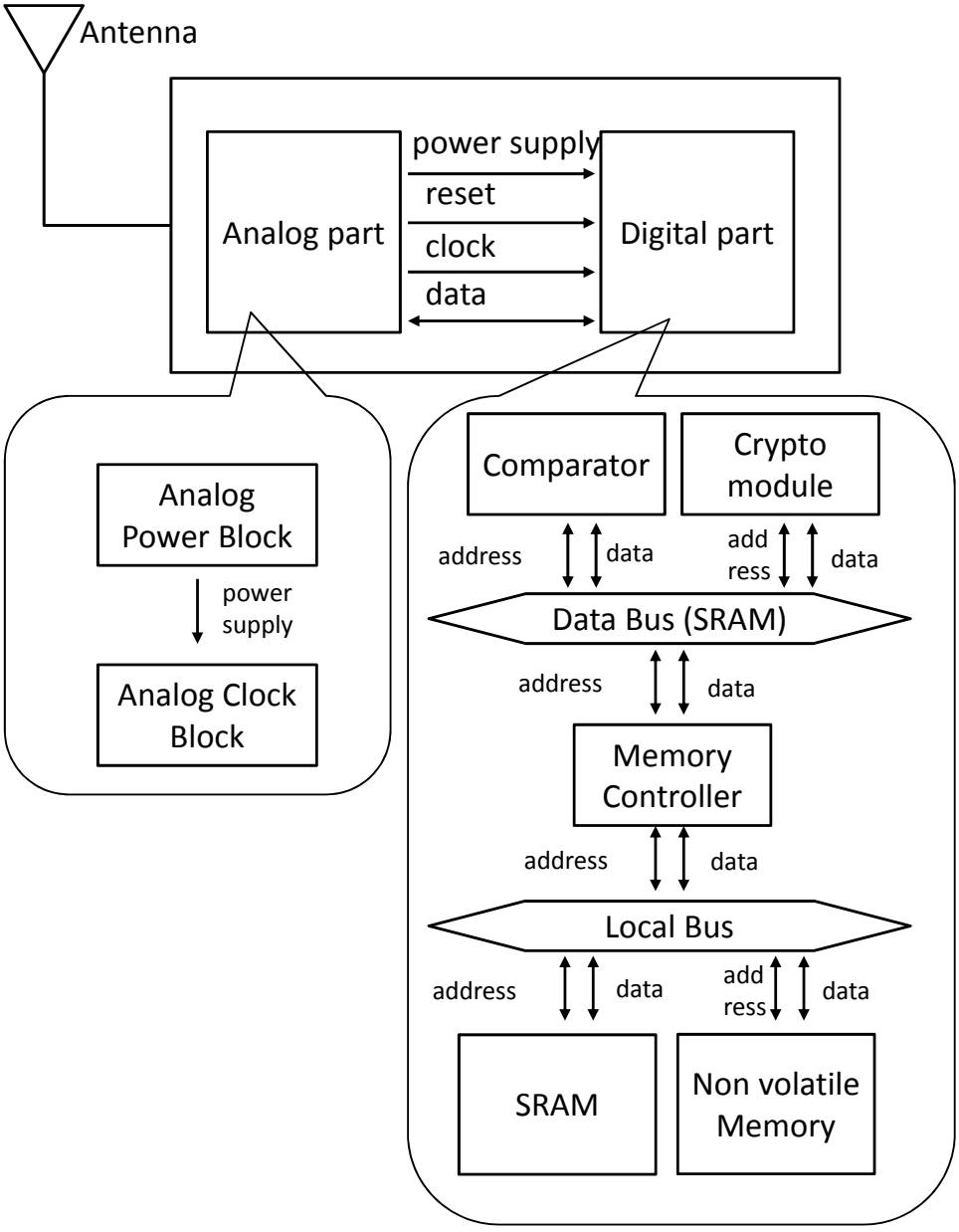


Figure. 5.1 Simple block diagram of RFID tag

ized by a datapath width of 8-bits that is the smallest datapath width to the best of our knowledge. If the datapath width is large, the power consumption of the tag is predicted to be large due to increase of active signals.

5.2.2 Interface of the Cryptographic Circuit

Since the cryptographic circuit is connected to SRAM, we design the interface of the cryptographic circuit based on the SRAM parameterized by a datapath width of 8-bit. Namely, key, IV, and message are cut into 8-bit blocks and the blocks are taken as the input to the cryptographic circuit one by one. One block is taken as the input to the cryptographic circuit within one clock cycle. Similarly, the 8-bit pseudorandom numbers or 8-bit hash value are returned from the cryptographic circuit within one clock cycle.

5.3 Implementation of the Lightweight Stream Cipher

5.3.1 Grain

5.3.1.1 Parallel Architecture

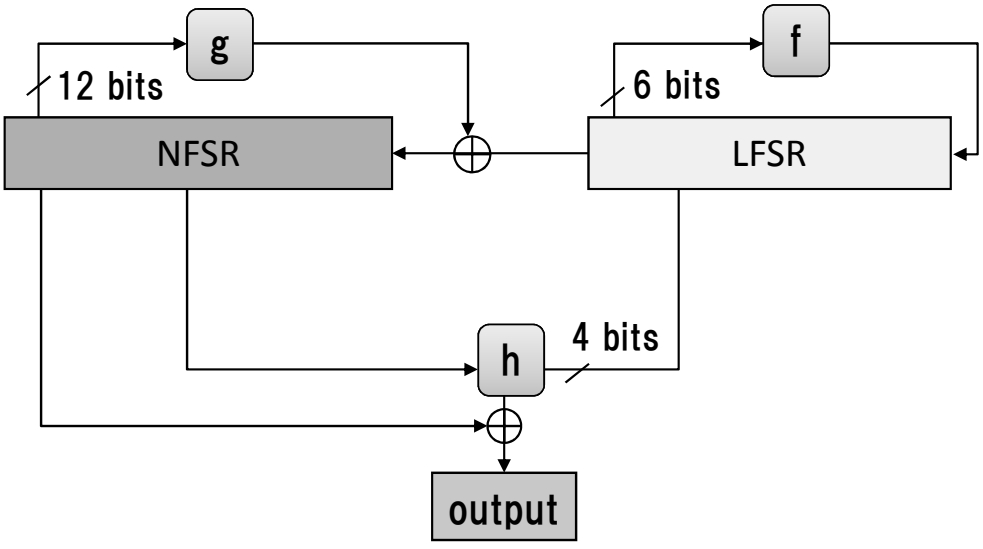


Figure. 5.2 Hardware architecture of Grain based on parallel architecture

Figure. 5.2 shows the hardware architecture of Grain based on the parallel architecture. The main part consists of registers and update functions. A 160-bit register is used to store 160-bit internal state. The register consists of a NFSR and a LFSR. The size of NFSR is 80-bit and the size of LFSR is 80-bit. The update function consists of f, g, and h functions. f function consists of XOR operations. g and h functions consist of AND and XOR operations. NFSR is updated with a value that is generated by xoring the outputs of g function and LFSR. LFSR is updated with the output of f function.

At the beginning of the initialization process, NFSR is initialized with key, and LFSR is initialized with IV and constants. Then, the registers are updated with the output of the update functions. In the initialization process, pseudorandom numbers are not returned.

After the initialization process, the pseudorandom number generating process begins. In this process, the output of h function is returned as pseudorandom numbers, and the internal state is continuously updated with the update functions.

5.3.1.2 Sequential Architecture

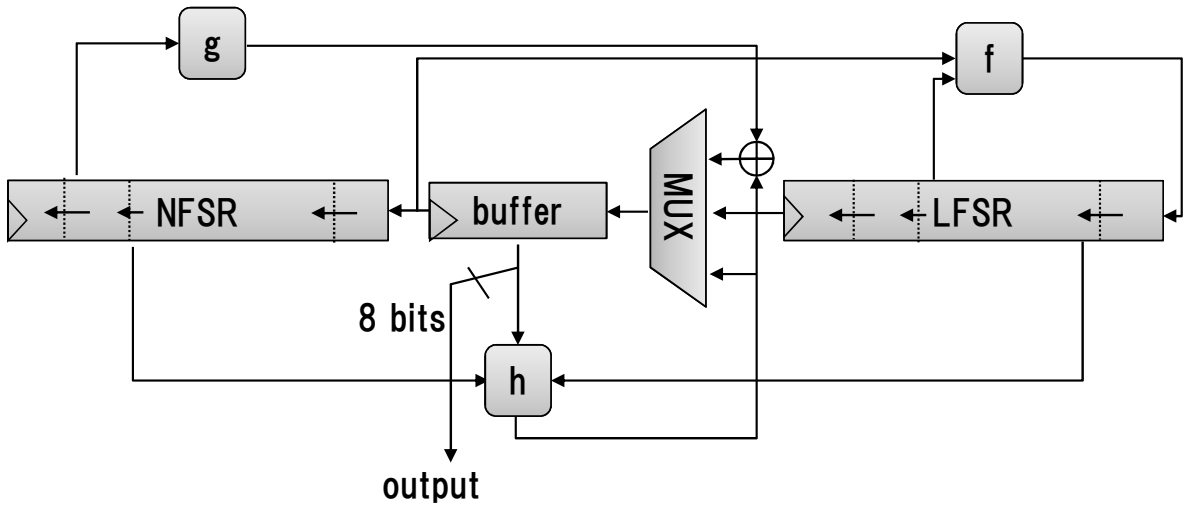


Figure. 5.3 Hardware architecture of Grain based on sequential architecture

Figure. 5.3 shows the hardware architecture of Grain based on the sequential architecture. We describe the update steps of the internal state in the initialization process. LFSR is represented as $(s_0, s_1, \ldots, s_{79})$. NFSR is represented as $(b_0, b_1, \ldots, b_{79})$.

1. Move data from $(s_0, s_1, \ldots, s_{15})$ to the buffer.
2. LFSR shifts 16-bit blocks one by one per clock.
3. Update the least significant 16-bit $(s_{64}, s_{65}, \ldots, s_{79})$ with data generated by xoring LFSR, the buffer, and the output of h function.
4. Store the input data to NFSR $(s_{64}, s_{65}, \ldots, s_{79})$ in the buffer. The data is generated by xoring NFSR, the buffer, and the output of h function.
5. NFSR shifts 16-bit blocks one by one per clock.
6. Move data, which are stored in the buffer at Step.4., to the least significant 16-bit of NFSR $(b_{64}, b_{65}, \ldots, b_{79})$.

In the pseudorandom number generating process, 16-bit pseudorandom number is stored in the buffer and is output every 8-bit. After that, the same processes described above are done. Table 4.1 shows the designated counter in the control circuit for Grain.

5.3.2 Trivium

5.3.2.1 Parallel Architecture

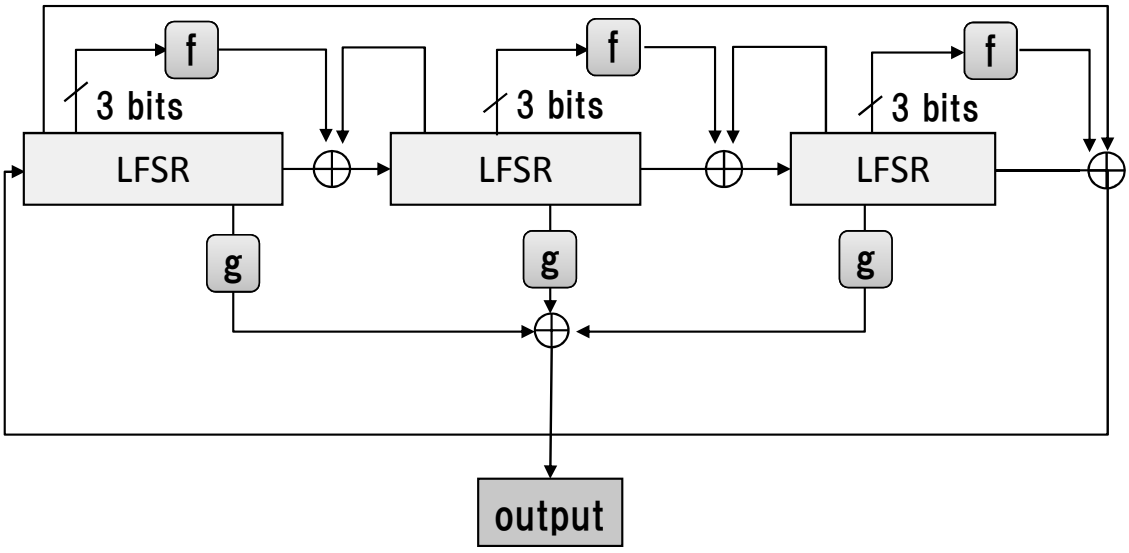


Figure. 5.4 Hardware architecture of Trivium based on parallel architecture

Figure. 5.4 shows the hardware architecture of Trivium based on the parallel architecture. The main part consists of registers and update functions. A 282-bit register is used to store 282-bit internal state. The register consists of three LFSRs. The sizes of LFSRs are 93-bit, 84-bit, and 111-bit, respectively. The update function consists of f and g functions. f function consists of XOR and AND operations. g function consists of XOR operations.

At the beginning of the initialization process, LFSRs are initialized with key, IV, and constants. Then, the registers are updated with the output of the update function. In the initialization process, pseudorandom numbers are not returned. After the initialization process, the pseudorandom number generating process begins. In this process, data generated by xoring the outputs of g functions is returned as pseudorandom numbers, and the internal state is continuously updated with the update functions.

5.3.2.2 Sequential Architecture

The internal state consist of three LFSRs having different bit lengths $(s_1, s_2, \ldots, s_{93})$, $(s_{94}, s_{95}, \ldots, s_{177})$, and $(s_{178}, s_{179}, \ldots, s_{288})$. We call these LFSRs $LFSR0$, $LFSR1$, and $LFSR2$ shortly. We show the hardware architecture of Trivium based on the sequential architecture in Figure. 5.5. Similar to Figure. 5.3, the initial input to LFSRs are not shown. We desribe the update steps.

1. Move data stored in $(s_{110}, \ldots, s_{125})$ to the buffer.
2. Shift $(s_{94}, \ldots, s_{109})$ 16-bit.
3. Update $(s_{94}, \ldots, s_{109})$.
4. Overwrite $(s_{78}, \ldots, s_{93})$ with $(s_{62}, \ldots, s_{77})$.

These steps are repeated in the order of $LFSR1$, $LFSR0$, and $LFSR2$. Finally, the date stored in the buffer are taken as the input to $(s_{126}, \ldots, s_{141})$.

In the pseudorandom number generating process, a 16-bit pseudorandom number is stored in the buffer and is output every 8-bit. After that, the same processes described above are done.

Table 5.1 shows the designated counter in the control circuit for Trivium. The counter has the charactaristic that the Hamming Distance between two continuous steps are always 1. As the internal state of the Trivium is larger than that of Grain, the number of steps are also larger.

Table 5.1 Counter for control circuit of Trivium

Round	Counter value
#	bit
0	00000
1	00001
2	00011
3	00111
4	00101
5	01101
6	01001
7	01011
8	01111
9	11111
10	10111
11	10011
12	11011
13	11001
14	11101
15	10101
16	10001
17	10000
18	10010
19	11010
20	01010
21	00010

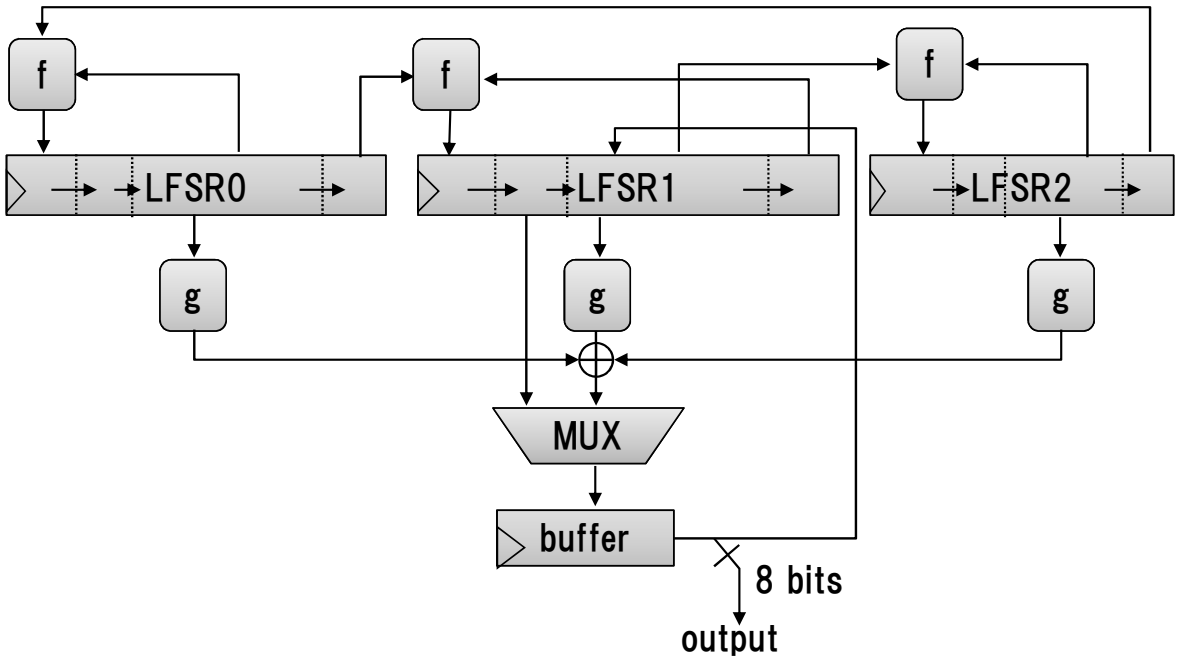


Figure. 5.5 Hardware architecture of Trivium based on sequential architecture

Therefore, we increase the bit width of the conter.

5.3.3 Enocoro

5.3.3.1 Parallel Architecture

Figure. 5.6 shows the hardware architecture of Enocoro based on the parallel architecture. The main part consists of registers and update function. A 176-bit register is used to store 176-bit internal state. The register consists of LFSR and FF. The sizes of LFSR and FF are 160-bit and 16-bit. LFSR is updated with a linear function that takes 8-bit of FF as an input. FF is updated with the sbox and the linear function. Sbox is 8-bit input and 8-bit output substitution function, and it takes 8-bit of LFSR as the input. The sbox consists of four 4-bit input and 4-bit output substitution function s4, and linear functions. The s4 block is implemented by a look-up table. The sbox block is implemented 4 times in parallel. The linear function is implemented using XOR and bit-wise shift operations.

At the beginning of the initialization process, LFSR and FF are initialized with key, IV, and con-

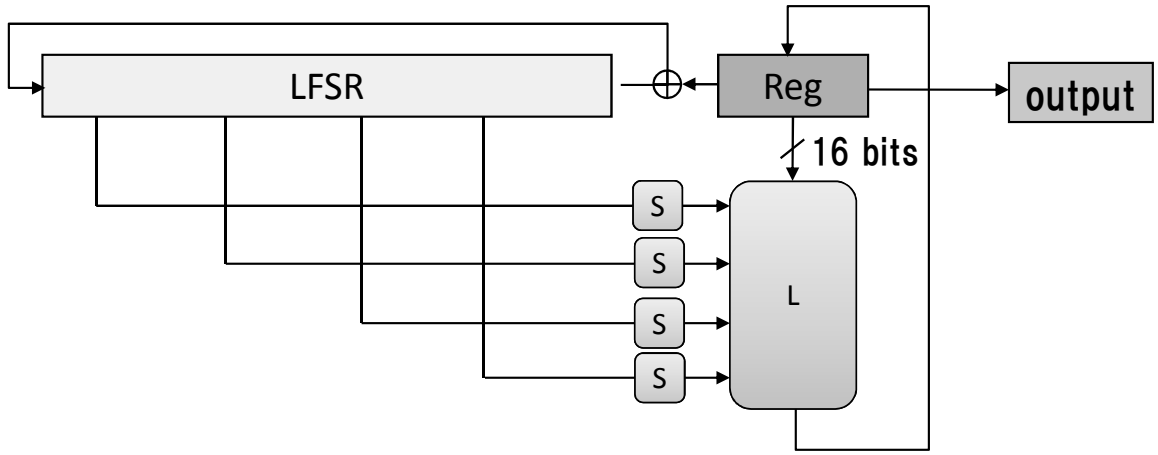


Figure. 5.6 Hardware architecture of Enocoro based on parallel architecture

stants. Then, the registers are updated with the output of the update function. In the initialization process, pseudorandom numbers are not returned. After the initialization process, the pseudorandom number generating process begins. In this process, 8-bit of FF are returned as pseudorandom numbers, and the internal state is continuously updated with the update function.

5.3.3.2 Sequential Architecture

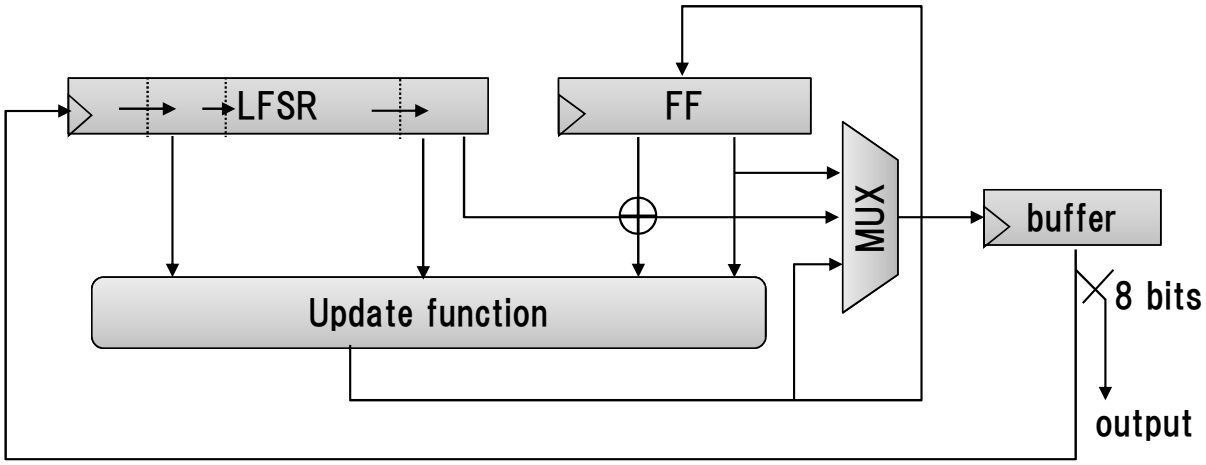


Figure. 5.7 Hardware architecture of Enocoro based on sequential architecture

Figure. 5.7 shows the hardware architecture of Enocoro based on the sequential architecture. We describe update sequence in the following. $(b_{19}, b_{18}, \ldots, b_0)$ and (a_1, a_0) indicate LFSR and FF, respectively.

1. Store the data generated by xoring b_{19}, b_{18}, and a_0 in the buffer.
2. Update (a_1, a_0)
3. $(b_{19}, b_{18}, \ldots, b_0)$ shifts 16-bit blocks one by one per clock.
4. Move the data stored in the buffer to b_0 and b_1

In the pseudorandom number generating phase, 16-bit pseudorandom number are stored in the buffer, and are output 8-bit per clock. After that, the same processes described above are done. Table 4.1 shows the designated counter in the control circuit for Enocoro.

5.4 Implementation of the Lightweight Hash Function

5.4.1 SPONGENT-160

Figure. 5.8 shows the hardware architecture of SPONGENT-160 based on the parallel architecture. A 16-bit buffer register is used to store output blocks. A 176-bit register is used to store 176-bit internal state. Data stored in the register is taken as the input to the round function. The round function consists of xor block, sbox block, and player block. The xor block xors 7-bit round constants and the 7-bit of the internal state. The sbox block substitutes the internal state by 4-bit input and 4-bit output sbox. The sbox is implemented by a look-up table. The sbox is implemented 44 times in parallel. The player function performs bit-wise permutation and is implemented with wires.

At the beginning of the initialization process, the registers are initialized with 0s. The register is updated with the output of the round function. In the squeezing process, hash value is produced. 16-bit of the register is returned as the hash block. Since an 8-bit datapath is used, the 16-bit hash block is stored in the buffer. Then, the hash block is output from the buffer. Two clock cycles are needed to output the hash block. The permutation process requires 90 clock cycles to absorb one message block or squeeze one hash value block.

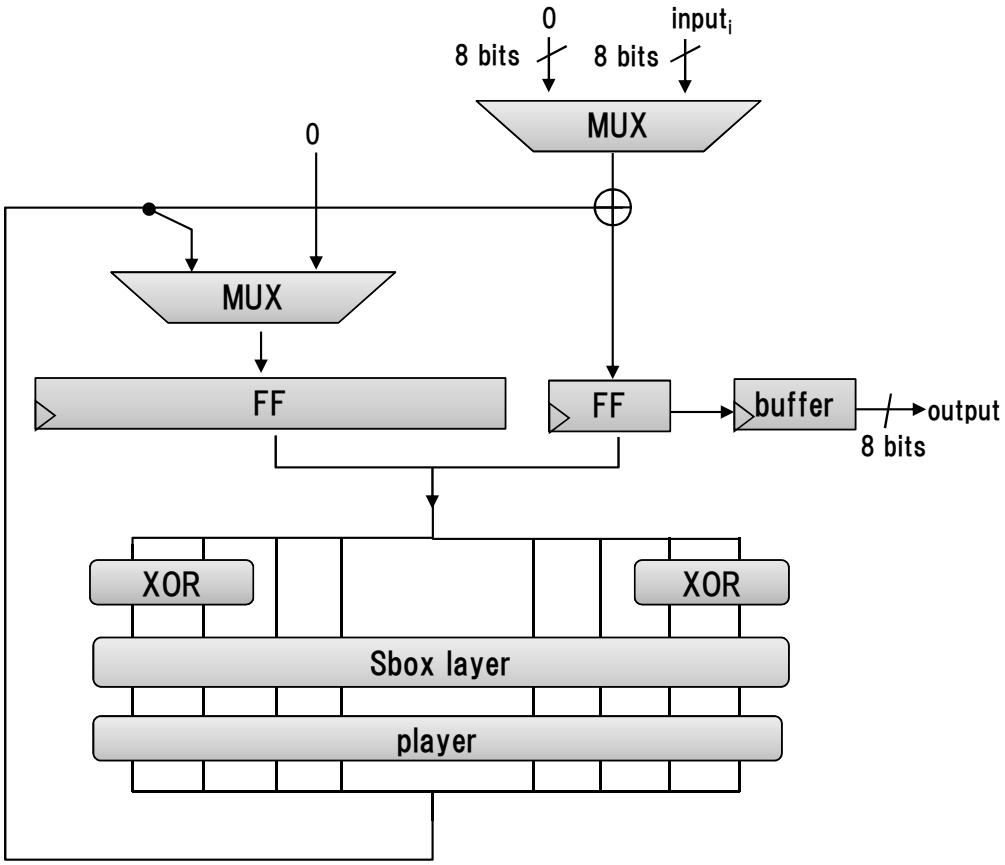


Figure. 5.8 Hardware architecture of SPONGENT-160 based on parallel architecture

5.4.2 D-QUARK

Figure. 5.9 shows the hardware architecture of D-QUARK based on the parallel architecture. The main part consists of buffer, register, and permutation function. A 16-bit buffer is used to store output blocks. A 176-bit register is used to store 176-bit internal state. The register consists of three parts i.e. two NFSRs and a LFSR. The size of NFSR is 80-bit and the size of LFSR is 16-bit. The permutation function consists of f, g, and h functions. These functions consist of AND and XOR operations. A NFSR is updated with a value generated by xoring the output of g and h functions. The other NFSR is updated with a value generated by xoring output of f and h functions. Data stored in the register is taken as an input to the round function.

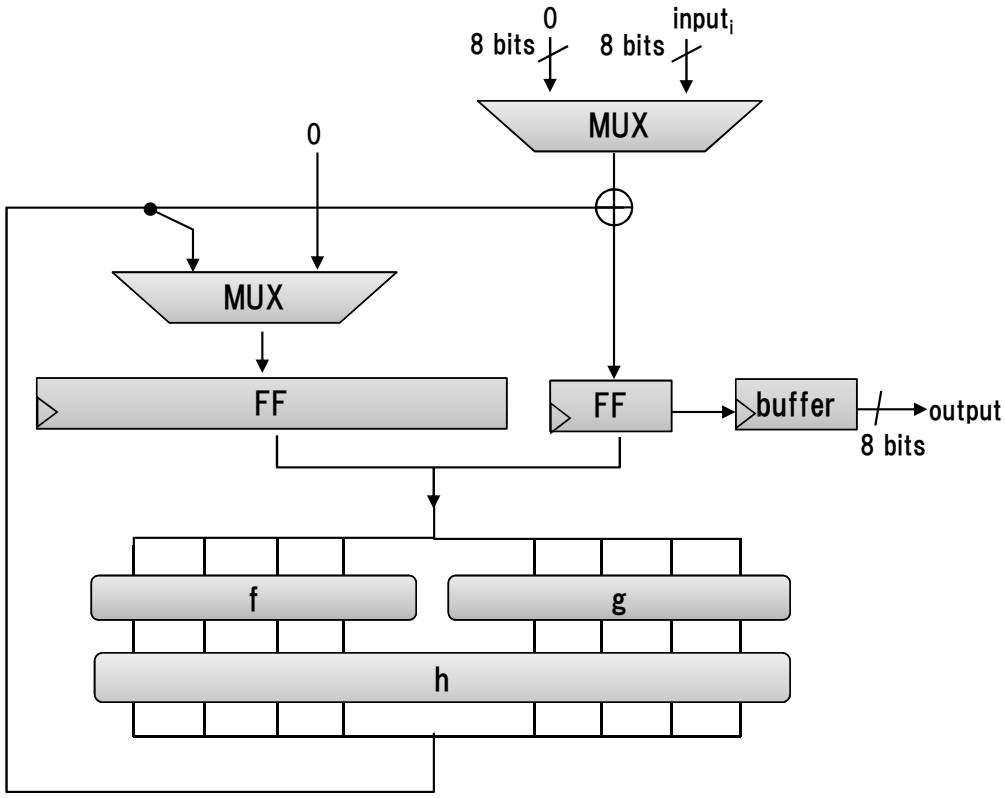


Figure. 5.9 Hardware architecture of D-QUARK based on parallel architecture

At the beginning of the initialization process, the registers are initialized with 0s. The register is updated with the output of the permutation. In the squeezing process, hash value is produced. 16-bit of the register are returned as the hash block. Since an 8-bit datapath is used, the hash block is stored in the buffer. Then the hash block is returned from the buffer. The permutation process requires 88 clock cycles to absorb one message block or squeeze one hash value block.

5.4.3 PHOTON-160/36/36

Figure. 5.10 shows the hardware architecture of PHOTON-160/36/36 based on the parallel architecture. The main part consists of buffer, register, and permutation function. A 36-bit buffer is used to store output blocks. A 196-bit register is used to store 196-bit internal state. Data stored in the register are taken as an input to the round function. The round function consists of add constant block, subcell block, shiftrow block, and mixcolumnserial block. The add constant block

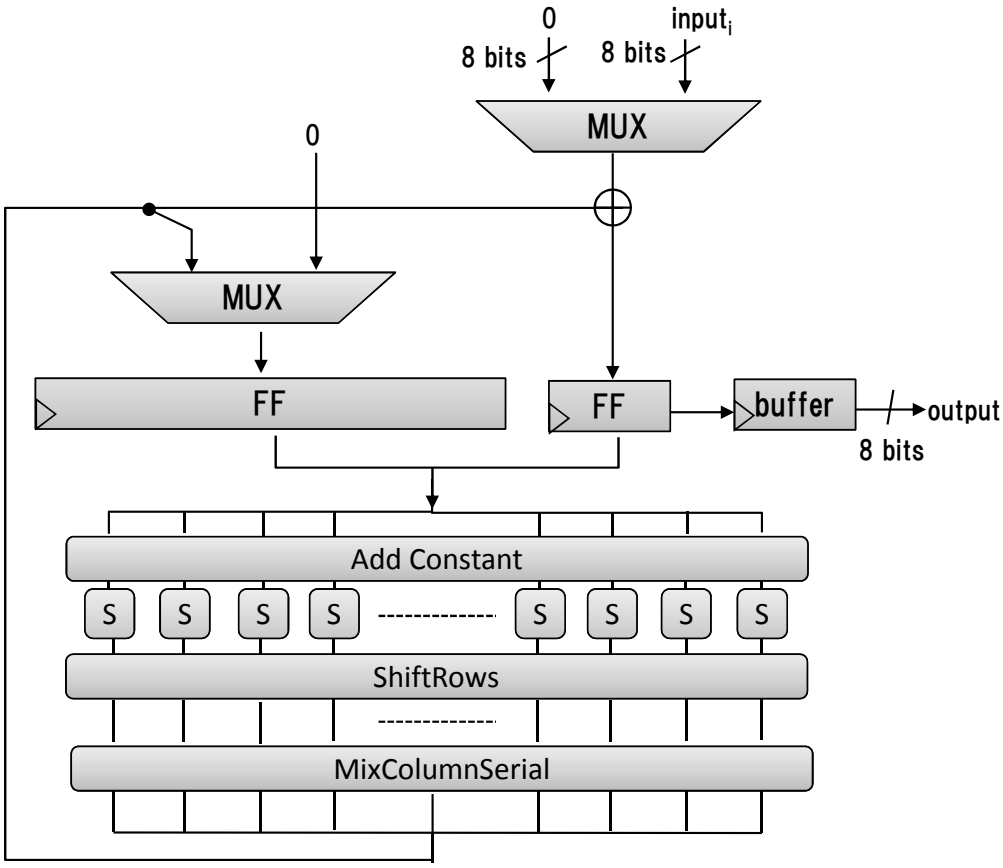


Figure. 5.10 Hardware architecture of PHOTON-160/36/36 based on parallel architecture

xors 4-bit round constant, 4-bit constant, and 4-bit state. The round constant block is implemented 5 times in parallel. The subcells block substitutes the state by 4-bit input and 4-bit output sbox. The sbox is implemented 49 times in parallel. The shiftrow block rotates all 4-bit states to the left. The mixcolumserial block updates all columns of the states linearly.

At the beginning of the initialization process, the registers are initialized with 0s. The register is updated with the output of the round function. In the squeezing process, hash value is produced. 36-bit of the register is returned as the hash block. Since an 8-bit datapath is used, the 36-bit hash value is stored in the buffer. Then the hash value is output from the buffer. Five clock cycles are needed to output the hash block. The permutation process requires 12 clock cycles to absorb one message block or squeeze one hash value block.

5.4.4 Keccak

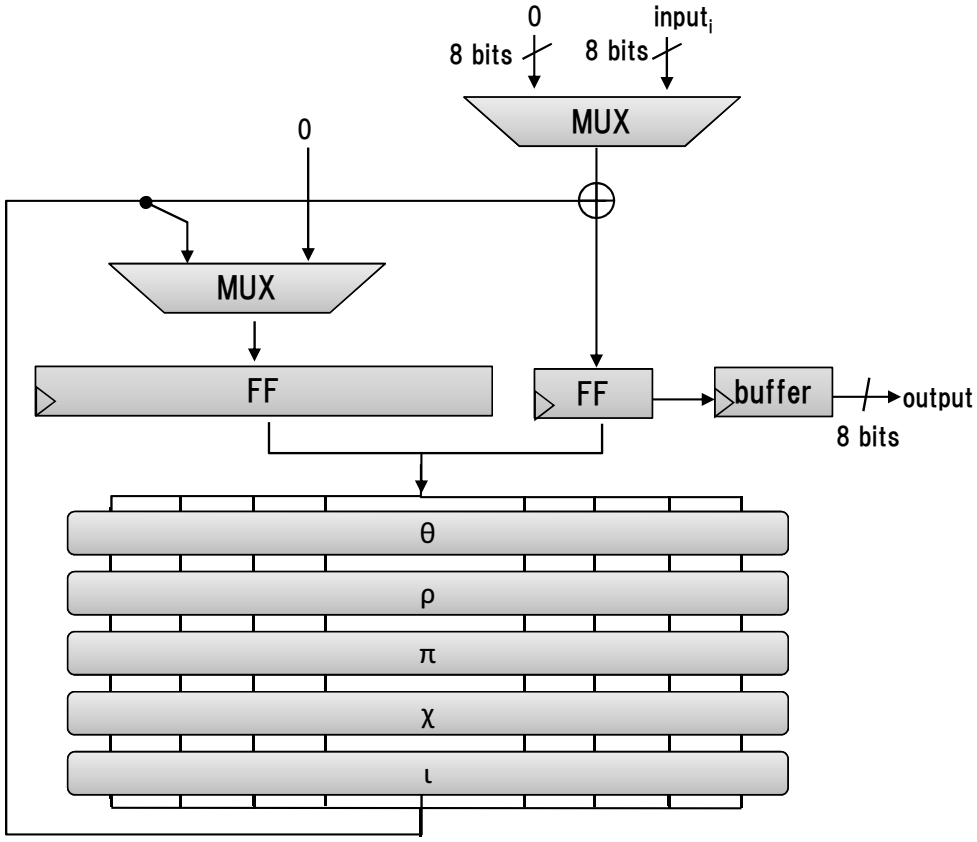


Figure. 5.11 Hardware architecture of Keccak based on parallel architecture

Figure. 5.11 shows the hardware architecture of Keccak based on the parallel architecture. The main part consists of register, buffer, and permutation function. A 1600-bit register is used to store the internal state. Data stored in the register is taken as an input to the round function.

The round function consists of θ block, ρ block, π block, χ block, and ι block. θ block consists of bit-wise permutation and XOR operation. θ block is implemented 64 times in parallel. ρ block consists of bit-wise permutation. π block consists of bit-wise permutation and XOR operation. π block is implemented 64 times in parallel. χ block consists of bit-wise AND, NOT, and XOR operations. χ block is implemented 64 times in parallel. ι block consists of XOR operations.

At the beginning of the initialization process, the registers are initialized with 0s. The register

is updated with the output of the round function. In the squeezing phase, hash value is produced. A part of the register is output as the hash block. Since an 8-bit datapath is used, the hash block is stored in the buffer and is output from the buffer. The permutation process requires 24 clock cycles to absorb a message block or squeeze a hash block.

5.5 Hardware Performance Evaluation

5.5.1 Performance Evaluation on FPGA

Our designs of the algorithms are described in Verilog-HDL. We have synthesized the circuits using Xilinx[*1] SPARTAN-6 FPGA XC6SLX45 device and Xilinx ISE 12.4.

Table 5.2 Hardware evaluation results of the lightweight cryptographic algorithms on FPGA

	Algorithms	Architecture	Area (Slice)	Time (Clock cycles)
Stream ciphers	Grain	Parallel	187	82
		Sequential	155	299
	Trivium	Parallel	223	167
		Sequential	153	1595
	Enocoro	Parallel	148	90
		Sequential	144	418
Hash functions	SPONGENT-160	Parallel	88	1534
	SPONGENT-160 [19]	Sequential	-	902880
	D-QUARK	Parallel	129	1500
	D-QUARK [3]	Sequential	-	160516
	PHOTON-160/36/36	Parallel	355	106
	PHOTON-160/36/36 [48]	Sequential	-	167842
	Keccak	Parallel	734	64
	Keccak [62]	Sequential	-	121240

Table 5.2 shows evaluation results of the lightweight stream ciphers and the lightweight hash functions on FPGA. We have not implemented the lightweight hash functions based on the se-

[*1] Xilinx, Spartan and ISE are registered trademarks of Xilinx, Inc.

quential architecture and the serialized architecture since these architectures seem not suitable for the tag from a viewpoint of the number of clock cycles.

5.5.2 Performance Evaluation on ASIC

We have synthesized the circuits using Synopsys[*2] Design Compiler[*3](Version C-2009.06-SP-5) and TSMC[*4] 90 nm process standard cell library. To evaluate the number of signal toggles, we have simulated the circuits' behavior for 100 random test vectors and have generated VCD files. We have calculated a population mean and a population standard by 99 % reliable section, and then we have predicted the number of signal toggles by three sigma limit.

Table 5.3 Hardware evaluation results of the lightweight cryptographic algorithms on ASIC

	Algorithms	Architecture	Area (GE)	Clock cycles	Signal toggle (♯ toggle)
Stream ciphers	Grain	Parallel	3470	82	108
		Sequential	3900	299	99
		Sequential (dedicated counter)	3850	299	43
	Trivium	Parallel	3810	167	110
		Sequential	4050	1595	42
		Sequential (dedicated counter)	4130	1595	32
	Enocoro	Parallel	3510	90	92
		Sequential	3640	418	46
		Sequential (dedicated counter)	3660	418	37
Hash functions	SPONGENT-160	Parallel	2980	1534	141
	D-QUARK	Parallel	3064	1500	182
	PHOTON-160/36/36	Parallel	7075	106	184
	Keccak	Parallel	32400	60	1192

Table 5.3 shows evaluation results of the lightweight stream ciphers and the lightweight hash functions on ASIC. We have not implemented the lightweight hash functions based on the sequen-

[*2] Synopsys is a registered trademark of Synopsys. Inc.

[*3] Design Compiler is a product name of Synopsys. Inc.

[*4] TSMC is a registered trademark of Taiwan Semiconductor Manufacturing Company Limited.

tial architecture and the serialized architecture since these architectures seem not suitable for the tag from a viewpoint of the number of clock cycles.

5.6 Discussion

We implement the lightweight stream ciphers with the sequential architecture, and then we evaluate hardware performance of these algorithms in ASIC. Our results show that the algorithms implemented with the sequential architecture archives up to 60 % lower number of signal toggles compared with the algorithms implemented with the parallel architecture.

In addition, we design a counter to reduce the glitch. The counter has a property that Hamming Distance between the continuous two values is always 1. We use the counter to implement these algorithms with the sequential architecture, and evaluate hardware performance in ASIC. Our results show that the algorithms implemented with the sequential architecture using our counter archives up to 60 % lower number of signal toggles compared with the algorithms implemented with the sequential architecture.

A cryptographic circuit consists of FF and combinational logic circuit. The update functions of the lightweight cryptographic algorithms consist of low-area consuming functions, such as XOR and bitwise rotations. Therefore, FFs take dominant part of the area requirements. Since the update function of PHOTON-160/36/36 consists of AES-based functions, the combinational logic circuit is more complicated. Thus, the contribution of the combinational logic of the algorithm is large. The internal state of Keccak is about 10 times larger than that of the lightweight cryptographic algorithms. As all bits of the internal state are taken as the input to the update function, the area requirements of the combinational logic circuit are large.

5.7 Conclusion

It is important to select a cryptographic algorithm taking into consideration of tradeoffs between hardware performances. When low-area requirements and low-power consumption are required, SPONGENT-160 is a good candidate. When high speed performance is required, Keccak can be used under the condition that the power consumption and the area requirements can be allowed.

The lightweight stream ciphers implemented with these hardware architectures can be used as these algorithms achieves well-balanced hardware performance.

Part IV

Design of Lightweight Pseudorandom Number Generator

Chapter 6

Lightweight Pseudorandum Number Generator

6.1 Introduction

PRNG is a key component for tag authentication protocols as well as cryptographic algorithms. Previous studies make it cleared that security weakness of the tags caused from mainly PRNG [68][45][78][49][44]. These results indicate that randomness of the output of PRNG is important for RFID authentication protocols.

PRNGs for tag have been studied. Lopez *et al.* have focused on Blum-Blum-Shub (BBS) PRNG and evaluated hardware performance on FPGA in the sense of area requirements and clock cycles [65]. Katti *et al.* have proposed a PRNG based on linear congruent generator using addition, multiplication, and modular reduction [60]. AKARI, which is a lightweight PRNG using addition and multiplication, has been developed, and hardware performance on ASIC have been studied [71]. Mujahid, *et al.* have proposed a PRNG using bitwise right rotation, bitwise OR, multiplication, and modulo addition [69]. They have evaluated hardware performance of their proposed method on FPGA. M-Segui *et al.* have proposed a PRNG based on LRSR configured with multiple polynomials that are selected by a physical source [94][95]. Chen *et al.* have proposed multiple polynomial LFSR based PRNG [27].

Since a tag is a small device, available hardware resources are quite limited. Using previously proposed PRNGs for a tag means that two kinds of circuits are needed to be implemented in a tag. One is a circuit for PRNG and the other is a circuit for the cryptographic algorithm. Therefore, more hardware resources are required if this implementation methodology is adopted.

We deduce that sharing a cryptographic circuit to generate pseudorandom number and ciphertext will achieve low area requirements of the tag. While PRNGs using a block cipher or a hash function are standardized by NIST [79], it is predicted that much hardware resources are required for implementation of NIST standardized PRNGs. These PRNGs cannot be realized by simply implementing the cryptographic algorithm as derivation functions are needed.

In this chapter, we study a method to generate pseudorandom number using a cryptographic algorithm that is a component of RFID authentication protocol. OMHSO protocol, which is a privacy preserving RFID authentication protocol, is based on PRNG and hash function. Thus, we focus on a pseudorandom number generation method using hash function for a tag.

6.2 Proposed Method

PRNG takes a seed, which is a secret data, as the input to the function and it returns pseudorandom number [73]. The value of seed changes every time for the authentication to generate different data. However, when we use a random number generator to generate the seed, additional hardware resources are required for the random number generator. As the output of the hash function changes drastically when the input to the hash function changes, we design a lightweight pseudorandom number generator based on hash chain.

Our proposed lightweight pseudorandom number generator works as follows. The seed is taken as the input to the hash function. The output of the hash function is divided into two parts. The first part is used for updated seed. The second part is used for pseudorandom number and this part is returned from the lightweight pseudorandom number generator. The following equations describe an overview of the lightweight pseudorandom number generator. Here, H is a hash function.

$$H(\text{Seed}) = \text{Updated Seed} \,||\, \text{Pseudorandom Number}, \tag{6.1}$$

$$\text{Seed} = \text{Updated Seed}. \tag{6.2}$$

This lightweight pseudorandom number generator enable for tags to generate pseudorandom number using the circuit of a hash function. This lightweight pseudorandom number generator does not require additional hardware components.

6.3 Evaluation

We evaluate the lightweight pseudorandom number generator from the viewpoint of the randomness and the security strength.

6.3.1 Experimental Evaluation

We have evaluated the randomness of the output of the lightweight pseudorandom number generator. We use NIST test suite [81] and Diehard test suite [30] since these test suites are widely used on previous studies. These test suites are statistical tests to measure the randomness of the data.

6.3.1.1 NIST Test Suite

NIST test suite consists of 15 tests.

1. Frequency Test

Test target is the proportion of zeroes and ones for the entire sequence. This test checks whether the number of ones and zeros in a sequence are approximately the same with the expected value in a random data.

2. Frequency Test within a Block

Test target is the proportion of ones within M-bit blocks. This test checks whether frequency of ones in an M-bit block is approximately M/2.

3. Runs Test

Test target is the total number of runs in the sequence. A run is an uninterrupted sequence of bits. The test checks whether the number of runs of ones and zeros of various lengths is different from the expected value in a random sequence.

4. Tests for the Longest-Run-of-Ones in a Block

Test target is the longest run of ones within M-bit blocks. The test checks whether the length of

the longest run of ones is different from the expected value in a random data.

5. Binary Matrix Rank Test

Test target is the rank of disjoint sub-matrices of the entire sequence. This test checks linear dependency among fixed length substrings of the sequence.

6. Discrete Fourier Transform (Spectral) Test

Test target is the peak heights in the Discrete Fourier Transform of the sequence. The test checks whether the value is different from the expected value in a random sequence.

7. Non-overlapping Template Matching Test

Test target is the number of occurrences of pre-specified target strings. The test checks production of many occurrences of a given non-periodic pattern.

8. Overlapping Template Matching Test

Test target is the number of occurrences of pre-specified target strings. In the Non-overlapping Template Matching Test, if the pattern is not found, the window slides one bit. On the other hand, in the overlapping Template Matching Test, if the pattern is found, the window slides one bit.

9. Maurer's "Universal Statistical" Test

Test target is the number of bits between matching patterns. The test checks whether the sequence can be significantly compressed without loss of information.

10. Linear Complexity Test

Test target is the length of LFSR. The test checks whether the sequence is enough considered as random.

11. Serial Test

Test target is the frequency of all possible overlapping m-bit patterns. The test checks whether the number of occurrences of the patterns is approximately same with the expected value in a random sequence.

12. Approximate Entropy Test

Test target is the frequency of all possible overlapping m-bit patterns. The test checks whether the frequency of overlapping blocks is the same with the expected value in a random sequence.

13. Cumulative Sums Test

Test target is the sum of digits, which is adjusted to 1 and -1, in the sequence. The test checks

whether the sum of the partial sequences is different from the expected value in a random sequence.

14. Random Excursions Test

Test target is the number of cycles in the sum of the partial sequences in a similar way in the Cumulative Sums Test. The test checks whether the number of cycles is different from the expected value in a random sequence.

15. Random Excursions Variant Test

Test target is the total number of times that a particular state visits the sum described in cumulative sum test. The test checks deviations from the expected value in a random number.

The results of the tests are analyzed in two ways. One is the evaluation of proportion of the blocks in the sequence that pass the test. The range of acceptable proportion is determined by three sigma limit. The other is the evaluation of distribution of p-values used to check the uniformity. A histogram is used on the evaluation. The interval between 0 and 1 is divided into 10 sub groups in the histogram. The p-values that lie in the sub groups are counted. Then, the uniformity of the p-values is determined by χ^2 test. When the significance level is under 1%, the test judges the sequence can be considered to be uniformly distributed. The results of two kinds of analysis are included in the output of the NIST test suite

6.3.1.2 Diehard Test Suite

Diehard test suite consists of 15 tests.

1. Birthday Spacing Test

This test chooses m birthdays in a year of n days and generates a list. The number of occurrence of the value in the list is expected to be in poisson distribution.

2. Overlapping 5-permutation Test

This test treats the input data as a sequence of random integers and determines the ordering of the consecutive five integers. Then, the number of each transition is counted. The variance of the transitions is expected to be a normal distribution.

3. Binary Rank Test (31×31, 32×32, 6×8)

This test creates random binary matrices. The sizes of the matrices are 31×31, 32×32, and 6×8.

The ranks of the matrices are calculated, and then the ranks are compared with expected value.

4. Bitstream Test

This test creates consecutive 20-bit letters and counts the number of missing of the letters which do not appear in the bit sequence. The numbers of missing of the letters are expected to be a normally distribution.

5. Overlapping Pairs Space Occupancy Test

This test creates consecutive 10-bit letters and counts the number of missing of the 2-letters which do not appear in the bit sequence. The numbers of missing of the letters are expected to be a normally distribution.

6. Overlapping Quadruples Sparse Occupancy Test

This test creates consecutive 5-bit letters and counts the number of missing of the 4-letters which do not appear in the bit sequence. The numbers of missing of the letters are expected to be a normally distribution.

7. DNA Test

This test creates 4-letters by 2-bit and counts the number of missing of the 4-letters which do not appear in the bit sequence. The numbers of missing of the letters are expected to be a normally distribution.

8. Count the 1s Test

This test focuses on the number of 1s appeared in the sequence. The sequences are generated by cutting input data into 8-bit integers, then, count the numbers of 1s in the bytes. The five kinds of letters are determined by the number of 1s in the bytes. Then, the distributions of the letters and the bytes are compared.

9. Parking Lot Test

This test focuses on randomly park a car in a square of side 100. If an attempt to park a car causes a crash, parking at a randomly chosen location is tried. This test counts the numbers of success to park a car. The numbers of success are expected to be a normally distribution.

10. Minimum Distance Test

This test chooses random points and finds the minimum distance between all pairs of the points. The square of the minimum distance is expected to be exponentially distributed.

11. 3D Spheres Test

This test chooses random points in a cube and finds the smallest value of the cube in the sense of radius. These values are expected to be exponentially distributed.

12. Squeeze Test

This test starts at $k = 2^{31}$ and finds number of iterations to reduce k to 1 by multiplying floating integer. The numbers of iterations are compared with expected value.

13. Overlapping Sums Test

This test counts the sum of consecutive 100 integers. The sums are predicted to be a normal distribution.

14. Runs Test

This test counts the number of up runs and down runs. Up runs mean that elements are increasing in value consecutively. Down runs mean that elements are decreasing in value consecutively.

15. Craps Test

This test plays crap game and finds the number of wins and the number of throws needed for end of the game. The numbers of win are predicted to be a normal distribution.

These tests return p-values. The Diehard test suite does not describe the level that the sequence can be considered to be uniformly distributed. Therefore, we judge the sequence in a similar way to the NIST test suite [100]. Namely, a histogram is used on the evaluation. The interval between 0 and 1 is divided into 10 sub groups in the histogram. The p-values that lie in the sub groups are counted. Then, the uniformity of the p-values is determined by using χ^2 test. Since the significance level of NIST test suite is 1%, we set the significance level as 1%. When the χ^2 value is less than 21.66, the sequence can be considered to be uniformly distributed

6.3.2 Data Setting

As these test suites require more than 10^6 binary sequence, we have generated 100 samples of 10^6 binary sequences using the lightweight pseudorandom number generator in accordance with the use on the tag authentication protocol. According to our previous studies, it has been cleared that the lightweight hash function SPONGENT-160 is a good candidate for a tag [77]. There-

fore, as a cryptographic component of the lightweight pseudorandom number generator, we select SPONGENT-160. We have generated a binary sequence according to the following steps. First, an 80-bit seed is taken as the input to the lightweight hash function. Then, the lightweight hash function generates 114-bit hash value and the hash value is divided into two parts. One is 80-bit seed and the other is 64-bit pseudorandom number. The pseudorandom number is returned as the output. The seed is used for the updated seed. These steps are repeated until desired number of binary sequence is generated. The binary sequence data for the tests are generated using a C program.

6.3.3 Evaluation Results on NIST Test Suite

The results are shown in Table 6.1.

Table 6.1 NIST test suite result summary

Test number	Test name	P-value (Pv)	Proportion (Pr)	Result
1	Frequency Test	0.851383	0.98	pass
2	Frequency Test within a Block	0.085587	1.00	pass
3	Runs Test	0.964295	1.00	pass
4	Tests for the Longest-Run-of-Ones in a Block	0.319084	1.00	pass
5	Binary Matrix Rank Test	0.983453	0.99	pass
6	Discrete Fourier Transform (Spectral) Test	0.514124	0.97	pass
7	Non-overlapping Template Matching Test	$0.005762 \leq Pv \leq 0.996335$	$0.96 \leq Pr \leq 0.99$	pass
8	Overlapping Template Matching Test	0.401199	0.99	pass
9	Maurer's "Universal Statistical" Test	0.739918	0.98	pass
10	Linear Complexity Test	0.262249	0.99	pass
11	Serial Test	0.834146, 0.739918	0.97, 1.00	pass
12	Approximate Entropy Test	0.062821	0.99	pass
13	Cumulative Sums Test	0.202268, 0.275709	0.98	pass
14	Random Excursions Test	$0.086458 \leq Pv \leq 0.900104$	$0.97 \leq Pr \leq 1.00$	pass
15	Random Excursions Variant Test	$0.001490 \leq Pv \leq 0.875539$	$0.97 \leq Pr \leq 1.00$	pass

6.3.4 Evaluation Results on Diehard Test Suite

The results are shown in Tables 6.2, 6.3, and 6.4. The distribution of p-value is shown in Table 6.3. χ^2 value of p-values and final result are shown in Table 6.4.

Table 6.2 Diehard test result summary

Test number	Test name	P-value (Pv)	Result
1	Birthday Spacing Test	$0.374339 \leq Pv \leq 0.943770$	pass
2	Overlapping 5-permutation Test	$0.065337 \leq Pv \leq 0.114126$	pass
3	Binary Rank Test	$0.050749 \leq Pv \leq 0.908538$	pass
4	Bitstream Test	$0.03576 \leq Pv \leq 0.94034$	pass
5	Overlapping Pairs Space Occupancy Test	$0.0098 \leq Pv \leq 0.9378$	pass
6	Overlapping Quadruples Sparse Occupancy Test	$0.0761 \leq Pv \leq 0.9841$	pass
7	DNA Test	$0.0548 \leq Pv \leq 0.9479$	pass
8	Count the 1's Test	$0.126291 \leq Pv \leq 0.997037$	pass
9	Parking Lot Test	$0.017844 \leq Pv \leq 0.962529$	pass
10	Minimum Distance Test	0.477304	pass
11	3D Spheres Test	$0.13528 \leq Pv \leq 0.93699$	pass
12	Squeeze Test	0.574709	pass
13	Overlapping Sums Test	$0.170076 \leq Pv \leq 0.978530$	pass
14	Runs Test	0.879693, 0.73519	pass
15	Craps Test	0.137325	pass

6.3.5 Security Analysis

We have evaluated the security strength of the lightweight pseudorandom number generator. We use the same settings that we have used in the randomness evaluation using NIST test suite and Diehard test suite. Let $seed_i$, H, and PRN_i be the ith seed, the hash function, and the pseudorandom number generated from $seed_i$, respectively. By introducing these symbols, the lightweight pseudorandom number generator is expressed by the following equation.

$$H(\text{seed}_i) = \text{seed}_{i+1} || \text{PRN}_i. \tag{6.3}$$

Table 6.3 Diehard test result summary

P-value range	Observed percentage	Expected percentage
0.0-0.1	2.8	10
0.1-0.2	12.03	10
0.2-0.3	8.8	10
0.3-0.4	8.8	10
0.4-0.5	12.97	10
0.5-0.6	10.65	10
0.6-0.7	9.72	10
0.7-0.8	14.35	10
0.8-0.9	8.8	10
0.9-1.0	11.11	10

Table 6.4 χ^2 value of p-values and final result

χ^2 value	Final result
19.462	pass

We evaluate the security strength of the lightweight pseudorandom number generator under the following assumptions. First, an attacker knows the pseudorandom number bits output from the RFID tag. Second, the tag has a tamper resistant memory and the attacker cannot get data stored in the memory. Third, the attacker has a simulator of the lightweight pseudorandom number generator.

We describe an attack scenario to reveal the secret seed of the lightweight pseudorandom number generator. First, we calculate the output of the hash function when we input all patterns of seeds by using the simulator. Since the bit length of the seed is 80-bit, the output patterns of the hash function are 2^{80}. Second, in the set of the output patterns of the hash function, we find $2^{80}/2^{64} = 2^{16}$ patterns having the same 64-bit sequence with PRN_i generated by the tag. This is because that the probability of the 64-bit data matching with PRN_i is $1/2^{64}$. These patterns can

be listed up since PRN_i is public. Third, we calculate the output of the hash function when we input above listed 2^{16} patterns in the second step to the hash function. Finally, in the set of the output patterns of the hash function, we search a pattern having the same sequence with PRN_{i+1} generated by the tag. When the pattern is found, a pattern of seeds selected in the first step is the secret seed.

Now we evaluate the computational complexity to reveal the secret seed on the above mentioned attack scenario. The computational complexity needed for the first step is 2^{80} since we calculate hash value when all input patterns are taken as the input to the hash function. The computational complexity needed for the third step is 2^{16} since we calculate hash value when 2^{16} patterns having the same sequence with PRN_i generated by the tag are taken as the input to the hash function. In the second step and final step, the hash function is not called. Therefore, total computational complexity to reveal the secret seed in the above mentioned attack scenario is approximately $2^{80} + 2^{16} \approx 2^{80}$. Thus, we deduce that the lightweight pseudorandom number generator supports 80-bit security strength under the condition that the seed is 80-bit and the pseudorandom number is 64-bit. This security strength is equal to the seed length.

6.3.6 Area Evaluation

According to the evaluation results of the lightweight cryptographic algorithms in the sense of area requirements in Chapter 5, the area requirements of the cryptographic circuit will be 6450 GE when SPONGENT-160 and Grain-80, which is intended to generate pseudorandom numbers, are implemented in the circuit. On the other hand, the area requirements of the cryptographic circuit will be 2980 GE when SPONGENT-160 is used to generate both hash values and pseudorandom numbers. Therefore, our proposed method achieves 53% less area requirements of the cryptographic circuit compared with the cryptographic circuit including a hash function and a PRNG.

6.4 Conclusion

We have studied a lightweight pseudorandom number generator for a tag. To achieve low-area requirements of the tag, we have designed a pseudorandom number generator based on a hash

function that is used as a component of a tag authentication protocol. Namely, first, a seed is taken as the input to a hash function. Then, the outputs of the hash function are divided into two parts. One is the updated seed and the other is the pseudorandom number. Finally, the pseudorandom numbers are returned from the lightweight pseudorandom number generator.

To check the randomness of the output of the lightweight pseudorandom number generator, we have used NIST test suite and Diehard test suite. A binary sequence is generated by the lightweight pseudorandom number generator under the following conditions. As the hash function, we have selected lightweight hash function SPONGENT-160. Seed length is 80-bit and length of pseudorandom number is 64-bit. The binary sequence generated by the lightweight pseudorandom number generator successfully passes all tests. Thus, we deduce that the output of the lightweight pseudorandom number generator can be thought uniformly distributed. We have also analyzed the security strength of the lightweight pseudorandom number generator and we deduce that the lightweight pseudorandom number generator achieves the same security strength with the seed size.

To generate pseudorandom number, some designated pseudorandom number generator have been proposed. NIST has announced PRNGs using a hash function or a block cipher. To realize the NIST method, additional components such as an expansion function are required to generate a long sequence of pseudorandom bits. On the other hand, our proposed method does not require additional components such as an expansion function, as the bit length of the pseudorandom number is not long for RFID tags. Therefore, our proposed method achieves low area requirements.

Part V

Design and Evaluation of Secure RFID Tag

Chapter 7

Fully Integrated Passive UHF RFID Tag for Hash-Based Mutual Authentication Protocol

Publication Data

Shugo Mikami, Dai Watanabe, Yang Li, and Kazuo Sakiyama, "Fully Integrated Passive UHF RFID Tag for Hash-Based Mutual Authentication Protocol," The Scientific World Journal, Hindawi, Volume 2015 (2015), Article ID 498610, 11 pages, Aug., 2015.

7.1 Introduction

To overcome privacy and security issues, privacy-preserving RFID authentication protocols based on cryptographic algorithms have been developed [90][2][99][31]. OSK protocol is one of the protocols based on hash functions [88][89]. An improved version of OSK protocol, which is called OMHSO protocol, has been developed [55].

Since cryptographic components, which form a building block of the protocols, increase power consumption and area requirements of the tag, hardware performance of the cryptographic mod-

ules or the digital processing block that runs a part of the authentication process based on cryptographic algorithms have been studied. ASIC implementation results of lightweight cryptographic algorithms have been discussed [16][58][43]. In [105], FPGA implementation results of the digital processing block that executes a hash-based challenge-response protocol have been studied. In [1], implementation results of OSK protocol on an IC card have been studied. In [36], ASIC implementation results of the digital processing block including AES [80] and Grain [7][54] have been studied. In [91], ASIC implementation results of CRYPTOGPS including lightweight block cipher PRESENT [21] have been studied.

In industry, some semiconductor manufacturers produce RFID tags with cryptographic components. ORIDAO produces a secured EPC Gen2 chip including 192-bit hash function [87]. NXP semiconductors sells MIFARE[*1] DESFire[*2] EV1 using block cipher 3DES [86].

In [72], Martin *et al.* have focused on two lightweight mutual authentication protocols. These protocols are based on a PRNG and simple functions such as rotation operation. They have designed a digital circuit including the PRNG, registers, and control logic. They have synthesized the circuit and evaluated hardware performance of the circuit in the sense of area requirements, power, and throughput. In [64], Liu *et al.* have proposed a lightweight mutual authentication protocol. The protocol requires a random number generator and a LFSR. They have designed a digital circuit including the LFSR and control logic. They have synthesized the circuit and evaluated hardware performance of the circuit in the sense of area requirements and power by performing a post-layout simulation.

They have implemented the digital circuit and have evaluated hardware performance of the digital circuit in the sense of area requirements and power consumption by performing a simulation. However, they have not carried out a silicon proof of the circuit and have not evaluated hardware performance of a tag including an antenna that is attached to the circuit to communicate over the air with a reader. In other words, feasibility of the protocol from the viewpoint of hardware implementation has not been cleared.

Implementation and evaluation of the whole tag, which runs the authentication process, have not

[*1] MIFARE is a registered trademark of NXP Semiconductors.

[*2] DESFire is a registered trademark of NXP Semiconductors.

been studied. An antenna, an analog front end, and a digital processing block should be integrated in the tag. It is very important to evaluate the whole tag since these modules are required to run the whole authentication process and these modules operate in a mutually coupled manner.

To run the authentication protocol on the tag, additional components such as an analog front end, SRAM, EEPROM, and an antenna are required. Therefore, we design a completely assembled tag including not only a digital circuit but also above mentioned components. We have carried out a full silicon proof of the tag. The silicon proof of the tag enables us to evaluate hardware performance of the tag in the sense of the maximum read distance and the operating time. These metrics have a strong impact on the usability of the tag.

[66] has introduced an implementation result of a passive UHF RFID tag chip that runs OMHSO protocol. The feasibility of the chip has been evaluated under the condition that the RF signal is generated by an RF signal generator. As the cryptographic component, SPONGENT-160 has been implemented.

7.2 Specification of a Fully Integrated Passive Tag

7.2.1 Overview

We implement a fully integrated passive UHF RFID tag that runs OMHSO protocol in this paper. We design a single chip including the RF front end and the digital processing block. If these two kinds of units are implemented separately on the tag, the power consumption of the tag is predicted to be large since an electric current is required to make to flow between the chips. To reduce the electric current between the analog part and the digital part, we have decided to integrate the analog front end and the digital processing block in a single chip. Our tag has the ability to communicate with a reader on the market without extra equipment. As the cryptographic component, we implement not only SPONGENT-160 but also Keccak.

7.2.2 Hash Function

We select SPONGENT-160 that achieves well-balanced performance in Chapter 5. NIST shows a guidance to migrate to the use of stronger cryptographic algorithms [82]. In the guidance, migration to 128-bit security strength is required. Thus, we also choose newly selected NIST standard hash function Keccak to support strong security strength.

To run the protocol on a tag, three kinds of hash functions and a random number generator are required. However, separate implementation of these components will result in large area cost of the tag and the tag will consume much power since the power consumption of the circuit is generally proportional to the area requirements of the circuit.

To reduce cost of the tag, we decided to use a minimum cryptographic component that performs the necessary functions. For three hash functions, we decided to use single hash function with different padding manners to realize different hash calculations. For a random number generator, we use the lightweight pseudorandom number generator in Chapter 6. Therefore, we only implement single hash function and use it for three hash value calculations and a pseudorandom number generation.

7.2.3 Memory

Memory is an important component for implementation of the tag.

7.2.3.1 SRAM

Since SRAM cannot save the written data when the power supply is lost, SRAM is mainly used for the temporary storage of data. Two additional transistors called access transistors are used to control the access to a cell. The access to the cell is enabled by the word line that is connected to the access transistors. The cell is connected to the bit lines through the access transistors. The bit lines transfer data when both read access and write access occurs. When read access occurs, voltage level of the bit lines are driven high and low by the cell. When write access occurs, voltage level of the bit lines are driven high and low by data that is written to the cell.

7.2.3.2 EEPROM

The principle of EEPROM is based on a capacitor that stores electric charge for a long time. The main component of EEPROM is a field effect transistor. An additional gate exists between the control gate of the field effect transistor and the silicon. The gate is located near the carrier material and is not connected to external power supply. The gate, which is called floating gate, is charged or discharged from the carrier material using tunnel effect. Therefore, a large potential difference should exist between the gate and the carrier material.

Current between source and drain is controlled by the stored charge at the floating gate. When an electric current flows between the source and drain, 1 is stored in the cell. When an electric current does not flow, 0 is stored in the cell. When writing 0 or 1 to a cell, a high voltage is applied to the control gate to use tunnel effect.

The number of write operation is limited. As electrons are captured by tunneling effect with the write operation, electrons are gradually trapped on the floating gate. Since the trapped electrons generate the electric field in the floating gate, this electric field lowers the threshold voltage between 0 and 1. After sufficient number of write operation, the difference between 0 and 1 becomes too small to identify. Thus, read operation failure occurs.

7.2.4 Implementation Details

In this section, we describe the functionality and the characteristics of the blocks in the tag chip. Figure. 7.1 shows the functional blocks of the single chip. We stress here that while we have implemented SPONGENT-160 only in the hash function block in [66], we implement both SPONGENT-160 and Keccak in the hash function block.

The tag chip consists of the analog part and the digital part. The analog part consists of the analog power block and the analog clock block. The analog power block extracts the energy from the RF field and supplies the power for all the rest of the blocks. This block has buffer capacitors. This block generates two supply voltages of 1.8 V and 3.3 V.

The analog clock block generates clock signals needed in the digital processing block. This block has the oscillator and generates 12.8 MHz clock signal that is the source of all the clock

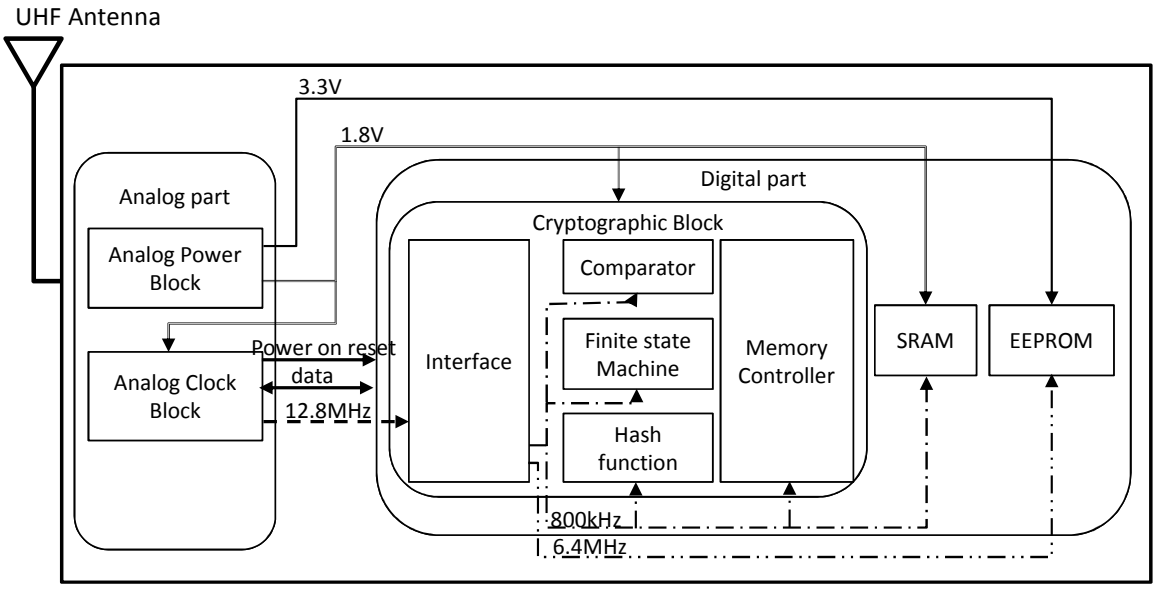


Figure. 7.1 Block diagram of the single chip

signals. This block also performs the demodulation/modulation and power on reset. The input supply voltage of this block is 1.8 V.

The digital part consists of the interface block, the comparator block, the finite state machine block, the hash function block, the memory controller block, the volatile memory, and the non-volatile memory. The interface block generates the clock signals from 12.8 MHz clock signal that is taken as the input to the block. The circuit in this block allows for lowering the clock frequency down to 800 kHz, 6.4 MHz, and 40 kHz. The input supply voltage of this block is 1.8 V. In addition, the interface block controls the communication between the analog part and the digital part. The response is generated in a module in this block. In the module, 40 kHz clock signal is used to generate the response. To simplify Figure. 7.1, this clock signal is not shown.

The comparator block compares the data such as the output of the hash function block, and the data sent from the server. This block operates with 800 kHz clock signal and the input supply voltage is 1.8 V.

The finite state machine block controls the flow of OMHSO protocol. The flow consists of the following 8 steps: loading tag state and seed for PRNG from EEPROM to SRAM, loading key from EEPROM to SRAM, calculation of α and β with hash function, calculation of Z' with hash function, comparing Z and Z', calculation of new key with hash function, EEPROM block writing for new key, and EEPROM block writing for new seed. This block operates with 800 kHz clock signal and the input supply voltage is 1.8 V.

The volatile memory is a 128-byte SRAM and is used to save temporal data such as the data received from the server, the secret data loaded from the non-volatile memory, and the output of the hash function block. The data width of SRAM is 8-bit to achieve low-power consumption of the tag. This block operates with 800 kHz clock signal and the input supply voltage is 1.8 V.

The hash function block calculates the hash values of the input data. The hash functions implemented on this block are SPONGENT-160 and Keccak. These algorithms are switched and used. This block operates with 800 kHz clock signal and the input supply voltage is 1.8 V.

The interface of the hash function module is based on SRAM with 8-bit data width. Namely, the input data is cut into 8-bit blocks and the blocks are taken as an input to the hash function module one by one. Similarly, hash value is cut into 8-bit blocks and the blocks are returned from the hash function module one by one.

The hash function module has FFs called state to store the intermediate data. After the state is initialized with 0, the input blocks are xored into a part of the state interleaved with the application of the permutation function of SPONGENT-160 or Keccak. At the end of every clock, the state is updated with the output of the permutation function. After absorbing all the input blocks, hash value is produced. A part of the state is returned as the hash block interleaved with the application of the permutation function. The state is updated continuously by the permutation function until desired bits of the hash value are returned.

The memory controller block communicates with SRAM and EEPROM for the data transfer. This block controls read/write operands, the address, and the data. This block operates with 800 kHz clock signal and the input supply voltage is 1.8 V.

Finally, the non-volatile memory is a 1-KByte EEPROM and is used to save the secret keys and the seed. The input supply voltage is 3.3 V. This block operates with 6.4 MHz clock signal.

Additional information about the blocks except the hash function block is described in [66].

7.2.5 Hardware Architecture of SPONGENT-160

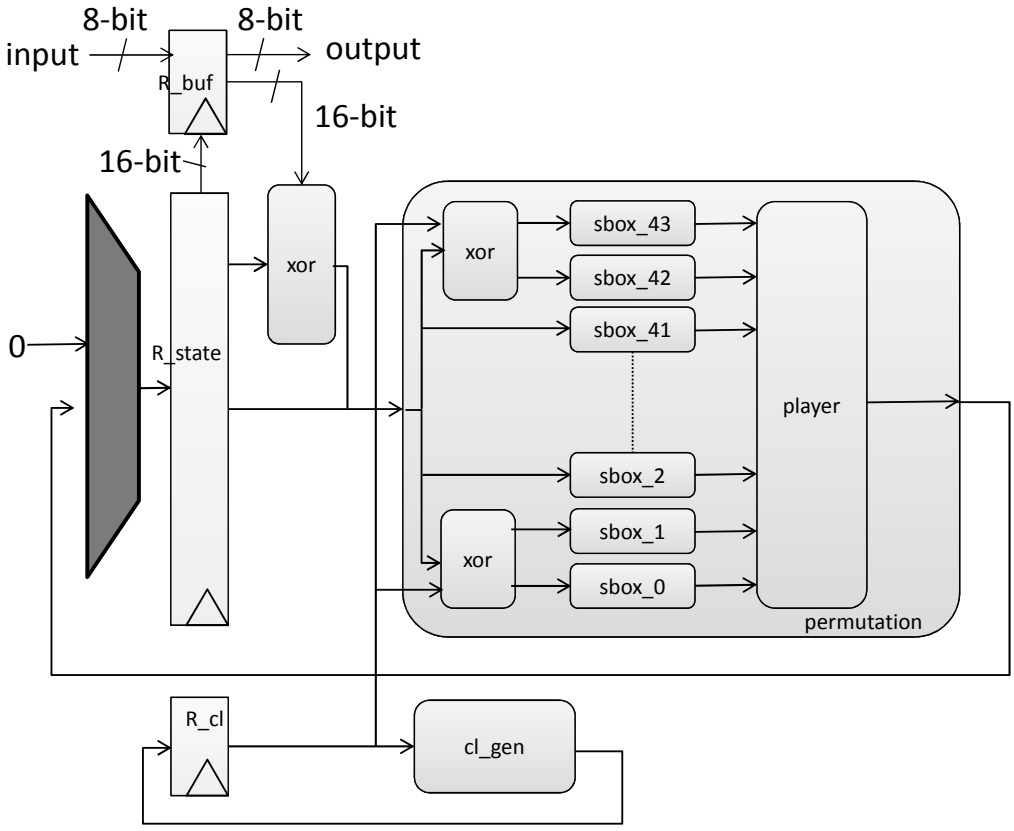


Figure. 7.2 Hardware architecutre of SPONGENT160 in the hash function block

Figure. 7.2 shows the hardware architecture of SPONGENT-160 in the hash function block. R-buf is a 16-bit buffer register to store message blocks and hash value blocks. R-state is a 176-bit register to save the internal state. Data stored in R-state is taken as an input to the permutation. R-cl is a 7-bit register to update the round constant.

At the beginning of the initialization process, the registers are initialized with 0. After a 16-bit message block is loaded to R-buf, the absorbing process begins. At the end of every clock in the absorbing process, R-state register is updated with the output of the permutation. In the squeezing

process, hash value is produced. 16-bit of R-state is output as the hash block. Since an 8-bit data path is used, the 16-bit hash block is stored in R-buf. Then the hash value is output from R-buf. Two clock cycles are needed to output the hash block.

7.2.6 Hardware Architecture of Keccak

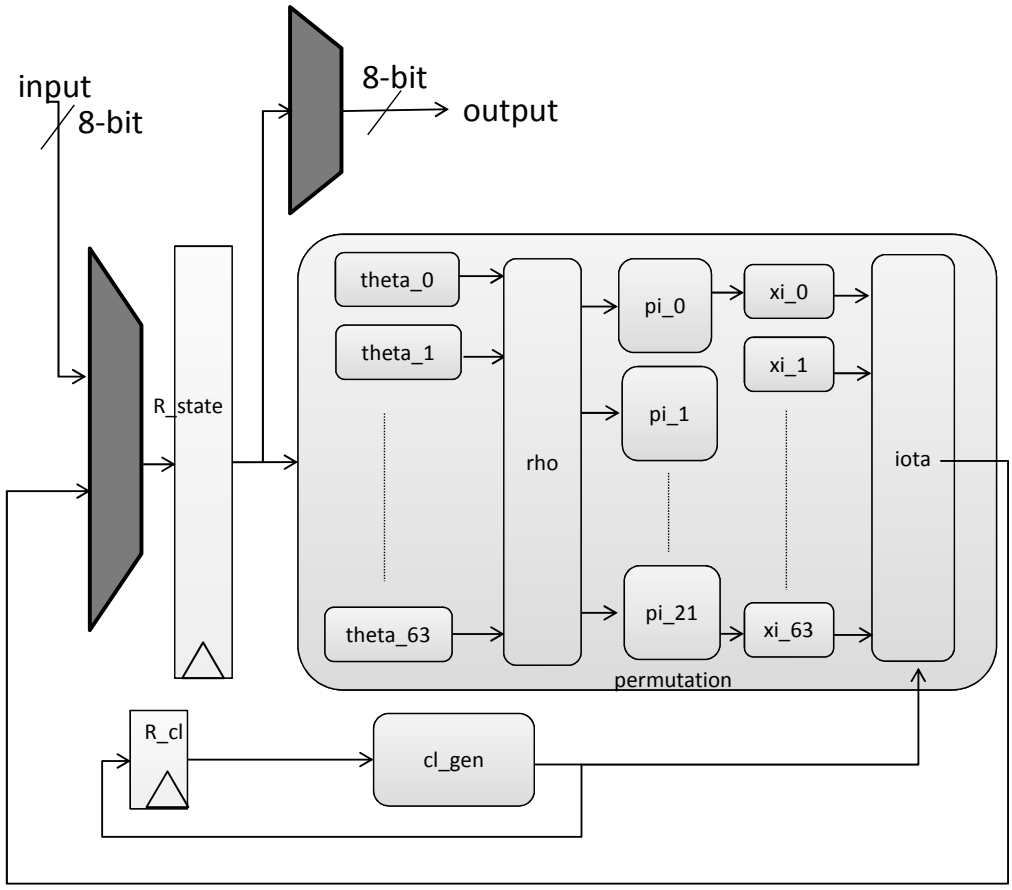


Figure. 7.3 Hardware architecture of Keccak in the hash function block

Figure. 7.3 shows the hardware architecture of Keccak in the hash function block. R-state is a 1600-bit register to save the internal state. Data stored in R-state is taken as an input to the permutation. R-cl is a 7-bit register to update the round constant.

At the beginning of the initialization process, the registers are initialized with 0. After a message

block is loaded to R-state, the absorbing process begins. At the end of every clock in the absorbing process, R-state register is updated with the output of the permutation. In the squeezing phase, hash value is produced. A part of R-state is output as the hash value. Since an 8-bit data path is used, the hash value is stored in the R-state and is output from the R-state.

Compared with the hardware architecture of SPONGENT-160, the hardware architecture of Keccak does not have R-buf. This difference is caused by the size of the rate. Since SPONGENT-160 has 16-bit rate, the number of the message block and the hash value block are large. Therefore, the permutation is applied to the internal state repeatedly interleaved with message block inputs and hash block outputs. On the other hand, since Keccak has more than 160-bit rate, the permutation is applied to the internal state only once to generate 160-bit hash value. Thus, the R-state is used to store the hash value after the permutation.

7.3 Hardware Evaluation Using the Actual Tag

In this section, we show evaluation results of the tag in terms of cost, speed, power, and the communication distance. As two hash functions can be switched and used, this mechanism enables us to evaluate the performance of the tag depending on the cryptographic hash functions.

7.3.1 Cost Evaluation

The whole RFID tag chip is fabricated in a 180 μm CMOS process. Figure. 7.4 shows the photograph of the manufactured RFID tag chip. As shown in Figure. 7.4, the tag chip includes the analog power block, the analog clock block, the cryptographic block, 128-byte SRAM, and 1-Kbyte EEPROM. The RFID tag chip size is 2.5 mm $\times$ 2.5 mm. The reason for selecting a relatively large chip size is due to the number of I/O pins required for the performance evaluation.

Table 7.1 shows the area requirements of the cryptographic block in the RFID tag chip. The results are based on the synthesis result and shown in both area and GE manner. The area requirements of the interface module are due to the functionalities of the clock divider, the reader command decoder, and the signal formation of the tag response. The area requirements of the cryptographic block occupied with SPONGENT-160 are 2.9 kGE. The area requirements of the

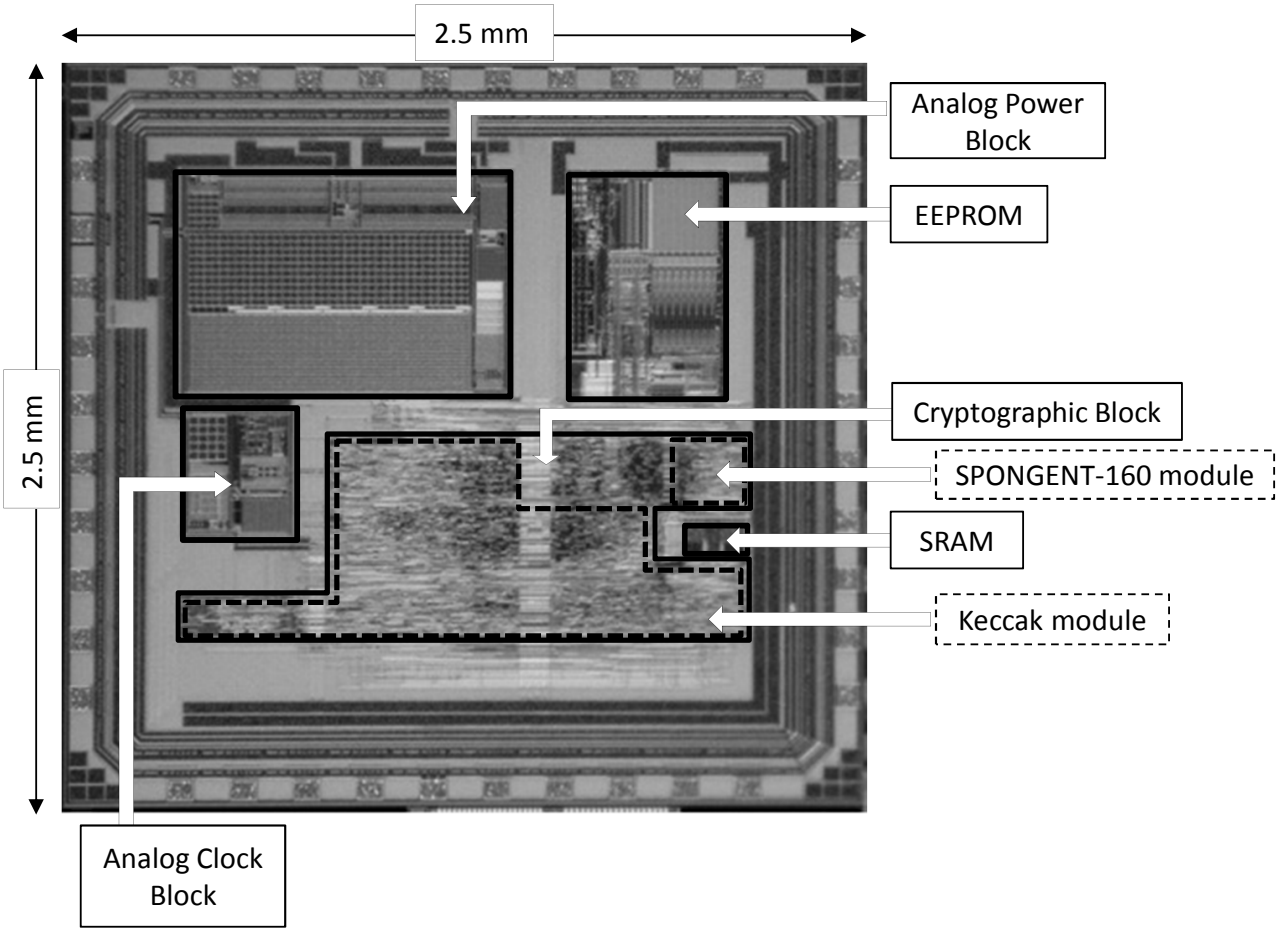


Figure. 7.4 Micrograph of the RFID tag chip

cryptographic block occupied with Keccak are 32.4 kGE.

Table 7.2 shows the area requirements of the each block in the layout of the RFID tag chip. The analog power block takes the largest portion of the area. Since the analog power block has buffer capacitors, these capacitors take a large portion of the area. EEPROM also takes a large portion of the area. However, EEPROM that we used has much larger memory size than the required memory size needed to run the authentication protocol.

Table 7.1 Area and GE of modules in the cryptographic block after synthesis

Component	Area (μm^2)	GE
Interface	50340	5734
SPONGENT-160 (Hash 1)	25809	2940
Keccak (Hash 2)	284641	32419
Others	11885	1353
Total	372675	42446

Table 7.2 Area of blocks on the RFID tag chip after layout

Component	Area (μm^2)
Analog Power Block	811915.6
Analog Clock Block	150955.7
Cryptographic Block	372675
Volatile Memory (SRAM)	39372.1
Non-Volatile Memory (EEPROM)	398950
Total	1773868.4

7.3.2 Speed Evaluation

Table 7.3 shows the operating time for each step of OMHSO protocol using SPONGENT-160 or Keccak. When SPONGENT-160 is used, the tag chip generates the first response to the reader within 5 ms. The tag chip finishes all the operations within 20 ms. The hash calculations and the EEPROM block write operations take large part of the time.

Similarly, when Keccak is used, the tag chip generates the first response to the reader within 0.3 ms. The tag chip finishes all the operations within 10 ms. The EEPROM block write operations take large part of the time.

Table 7.3 Time evaluation of OMHSO protocol using SPONGENT-160 or Keccak on the tag. The results using SPONGENT-160 are already mentioned in [66].

Step	Process	Time (ms)	Time (ms)	Difference of Time (ms)
	Hash function	SPONGENT-160	Keccak	-
(1)	Load tag state and seed for PRNG from EEPROM to SRAM	0.02	0.02	0
(2)	Load key from EEPROM to SRAM	0.02	0.02	0
(3)	Calculate α and β with Hash function	4.52	0.23	4.29
(4)	Calculate Z' with Hash function	2.94	0.14	2.8
(5)	Compare Z with Z'	0.06	0.06	0
(6)	Calculate new key with Hash function	2.04	0.11	1.9
(7)	EEPROM block write for new key	4.33	4.33	0
(8)	EEPROM block write for new seed	4.33	4.33	0
	Total	18.26	9.24	9.02

Next, we evaluate the time needed to run the whole process of OMHSO protocol including communication between the tag and a reader. The length of Tari is 25 μs in the wireless communication between the tag and the reader. The average data rate for the reader-to-tag communication is 27 kbps. The average data rate for the tag-to-reader communication is 95 kbps. The tag and the reader send 224-bit data each other. Thus the predicted time needed to exchange the data is 10.7 ms in total.

Therefore, the time required to complete OMHSO protocol using SPONGENT-160 is less than 30 ms. Similarly, the time required to complete OMHSO protocol using Keccak is less than 20 ms.

The tag using Keccak finishes authentication process about 9 ms faster than the tag using SPONGENT-160. Since the time required for the processes without hash calculation are independent from the hash functions, the difference of the time results from the hash calculations.

While the number of clock cycles needed for the cryptographic process of Keccak are smaller than that of SPONGENT-160, the cryptographic process of Keccak is more complex than that of SPONGENT-160. As shown in Section 7.3.1, the area requirements of SPONGENT-160 are smaller than that of Keccak. Therefore, there is a trade-off between the area requirements and the

time.

7.3.3 Power Evaluation

7.3.3.1 Simulated Power Consumption after Layout

We show the average power consumption of the cryptographic block based on a post-layout simulation. The baseband input signal is prepared in the simulation and is taken as the input to the cryptographic block. The post-layout netlist and the gates information are used to generate the signal toggle information. Based on these power consumption profiles, the average power consumption for each step of OMHSO protocol is evaluated.

Table 7.4 Simulated power consumption of the cryptographic block using SPONGENT-160 (μW)

Step	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Interface	157.0	155.0	154.0	154.0	153.0	153.0	153.0	153.0
SPONGENT-160	19.4	19.4	39.5	39.5	19.8	39.5	19.4	19.4
Keccak	1.1	1.1	1.1	1.1	1.2	1.1	1.1	1.1
Others	83.5	84.5	25.4	26.4	88.0	26.4	25.5	25.5
Total	261.0	260.0	220.0	221.0	262.0	220.0	199.0	199.0

Table 7.5 Simulated power consumption of the cryptographic block using Keccak (μW)

State	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Interface	148.0	146.0	147.0	147.0	147.0	148.0	145.0	144.0
SPONGENT-160	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Keccak	132.0	132.0	315.0	283.0	130.0	340.0	130.0	130.0
Others	76.9	77.9	48.9	80.9	80.9	47.9	18.9	19.9
Total	357.0	356.0	511.0	479.0	358.0	536.0	294.0	294.0

Table 7.4 shows the average power consumption of the cryptographic block using SPONGENT-160. Similarly, Table 7.5 shows the average power consumption of the cryptographic block using

Keccak. In Tables 7.4 and 7.5, step is the same as the one used on Table 7.3. In this simulation, the power consumption of SRAM and EEPROM is not included since the power consumption of these components cannot be evaluated correctly by the same method.

In Tables 7.4 and 7.5, the hash function module is active in Steps (3), (4), and (6). When the hash function module is active, the hash function module consumes about double or three times of power. The interface module, which includes the clock divider, constantly consumes about 150 μW power. The interface module takes the largest part of the power consumption of the cryptographic block when SPONGENT-160 is used. On the other hand, when Keccak is used, the hash function module takes the largest part of the power consumption of the cryptographic block.

7.3.3.2 Power Consumption of the Chip with RF-based Power Supply

In this section, we show evaluation results of the tag in terms of power consumption of the digital processing block including SRAM and EEPROM. We deduce that it is very important to use a single tag chip for evaluation since the performance of the tag chips are individually different. Therefore, we use the evaluation board of the chip to switch and use the hash functions on a single tag chip.

Next, we describe the evaluation methodology. Figure. 7.5 shows the evaluation system of the RFID tag chip using the evaluation board. The board has a debug functionality. The debug functionality enables us to observe the voltage via the debug pins. The functionality is set by a dip switch on the board.

There are two methods to supply power to the evaluation board: an external stable power supply and RF-based power supply. Originally, a passive tag harvests the necessary power from the RF field. To evaluate the performance of the tag under actual operating conditions, we decided to supply power to the chip from the RF field. We connect the antenna that is connected to an SMA connector on the evaluation board via cable. We attach the cable to the height adjustment device so that the distance between the antenna and the reader is variable. The data communicated between the reader and the tag are controlled on PC. The signal level is set to 24 dBm (250 mW) for the output of the reader.

For power evaluation of the digital processing block including SRAM, we use an oscilloscope to measure the voltage drop in the 1.8 V power supply line. We add several resistors in the 1.8

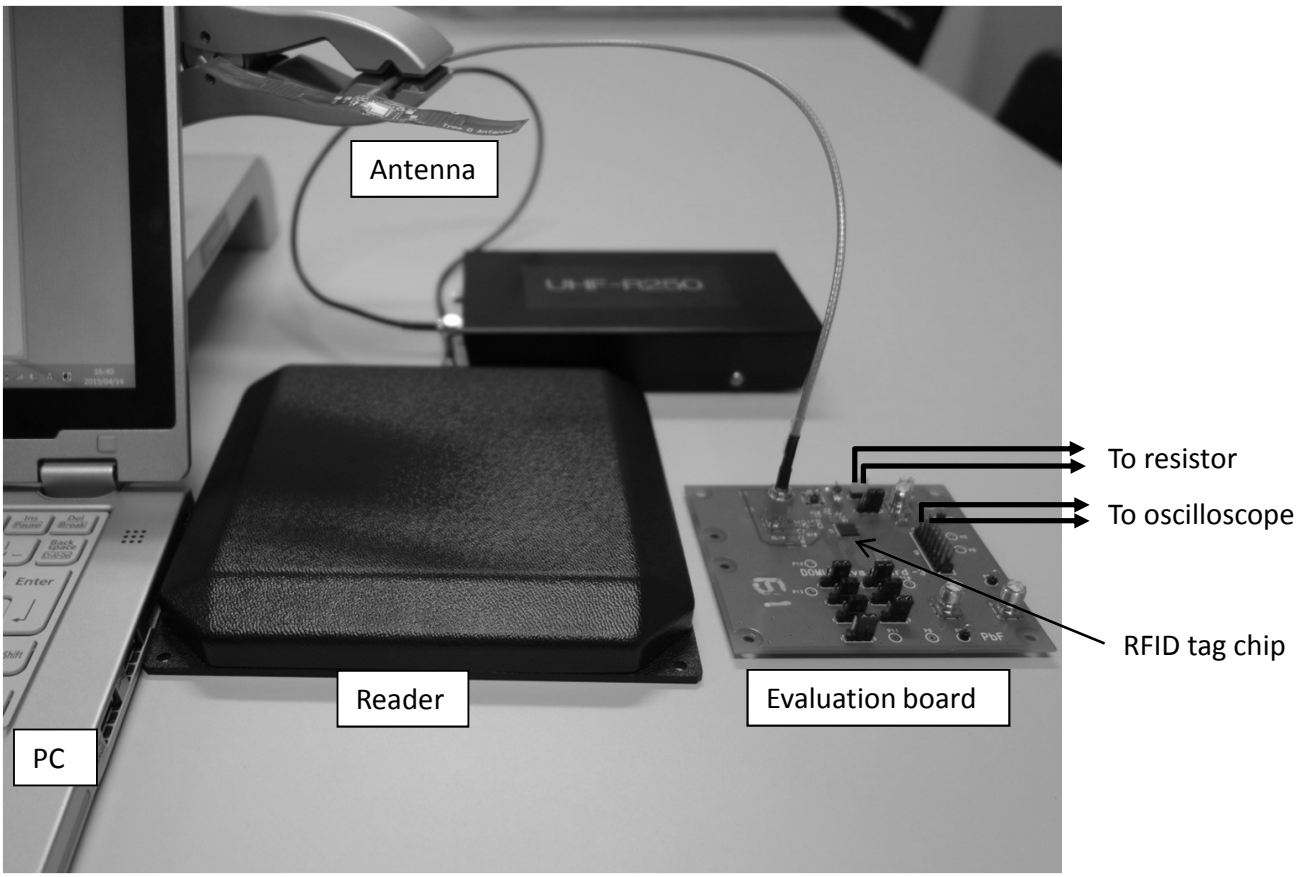


Figure. 7.5 Power evaluation system of RFID tag chip

V power supply line, which is connected to the digital processing block, to measure the voltage drop. Next, we calculate the electric current by dividing the voltage drop by the resistance. Then, we evaluate the power consumption of the digital processing block by multiplying the calculated electric current and the voltage after the resistor.

In Figure. 7.6, we plot the peak power consumption of the digital processing block using SPONGENT-160 versus the resistance. Similarly, Figure. 7.7 shows the peak power consumption of the digital processing block using Keccak versus the resistance.

Now we evaluate the real peak power consumption of the digital processing block with these figures. From these figures, it seems that there is a linear relationship between the power con-

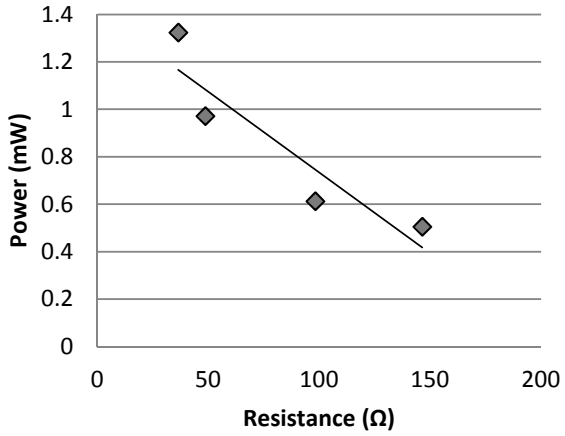


Figure. 7.6 Peak power consumption of the digital processing block using SPONGENT-160

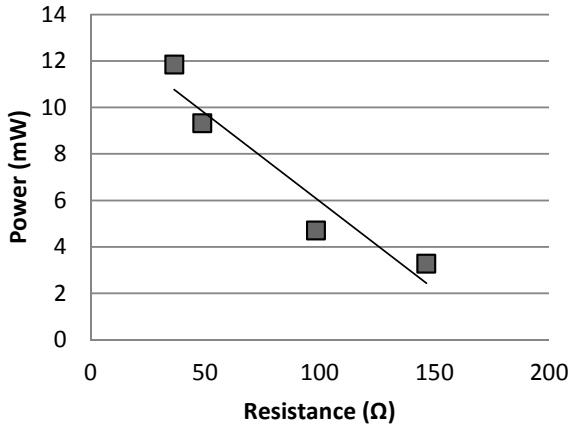


Figure. 7.7 Peak power consumption of the digital processing block using Keccak

sumption and the resistance. Usually the tag runs OMHSO protocol without the resistors. Thus, the actual power consumption of the block is the power that is measured under the condition that the added resistor will be 0 Ω. The actual values are equal to the intercepts of Figures. 7.6 and 7.7. The intercepts of Figures. 7.6 and 7.7 are 1.6 mW and 14.3 mW, respectively.

In the same way, we evaluate the power consumption of EEPROM during the authentication process is executed. In this case, we measure the voltage drop in the 3.3 V power supply line. The peak power consumption of EEPROM is 3.5 mW.

7.3.4 Communication Distance Evaluation

In this section, we show evaluation results of the maximum read distance between the tag and the reader. We first show our evaluation methodology. Similar to the power evaluation, we use the evaluation system shown in Figure. 7.5. On this evaluation, we turn off the debug functionality and remove the resistor added in the power supply line since there is no need to measure the voltage drop.

We first set the distance between the reader and the antenna. Then the tag runs OMHSO protocol 50 times repeatedly and we count the number of successes. To stabilize the wireless communication between a tag and a reader, a commercial reader sends the command repeatedly depending on the application or communication environment of the tag. Thus, we decided to judge whether the communication is successful or unsuccessful by sending the command several times. The success probability is calculated by dividing the number of the successes by the number of the execution of the protocol.

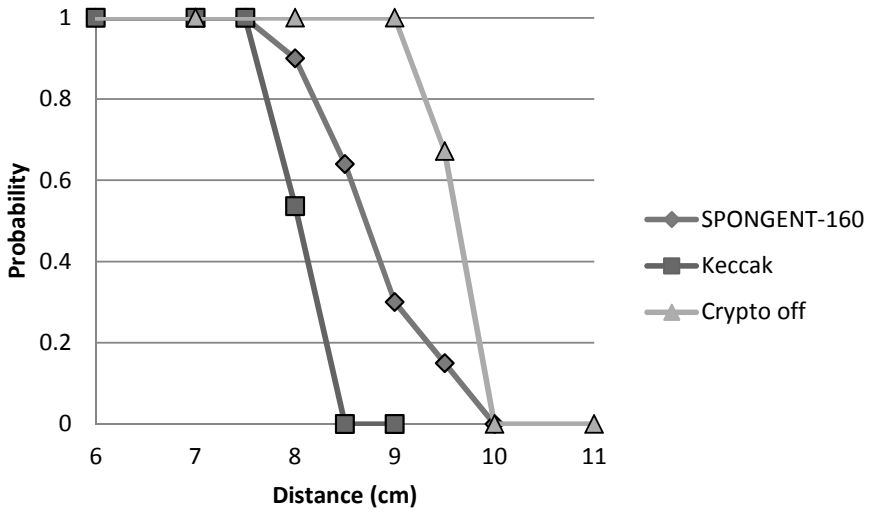


Figure. 7.8 The communication distance between the tag and a reader

We plot the communication distance and the success probability of the communication of each

hash function in Figure. 7.8. The maximum read distance of the tag using SPONGENT-160 is 1.5 cm longer than that of the tag using Keccak.

However, it remains the possibility of error in the measured maximum read distance according to the electromagnetic wave conditions such as the reflected waves. To evaluate the error, we repeatedly performed the above mentioned experiment. We focus on the distance where the success probability is 0 and it is close to 0. Our experimental results show that the number of the successes varies at the distance where the success probability is close to 0. On the other hand, the number of the successes are always 0 where the success probability is 0.

7.4 Discussion

7.4.1 Impact of Non-Volatile Memory

As described in Section 7.3.3, the peak power consumption of EEPROM is larger than that of the digital processing block using SPONGENT-160. Thus, when the lightweight hash function is used, the power consumption of EEPROM is a bottleneck.

Recently, low-power memories such as FRAM are being developed [92]. The principle of FRAM is based on the ferroelectric effort that is a capability of a material to retain an electrical polarization even if electric field is absent. The polarization is generated by the alignment of an electric dipole in the lattice of the ferroelectric material and is caused by the electric field that is stronger than the coercive force of the ferroelectric material. When an opposite electric field is applied to the ferroelectric material, the opposite alignment of an electric dipole is generated. The electric dipole takes one of two stable states and the stable states are saved after the electric field is removed. These states indicate 0 or 1.

FRAM is superior to EEPROM in terms of power consumption and speed performance. In the report [93], while EEPROM takes 5100 μs for write operation, FRAM takes 137 μs. While EEPROM consumes about 470 μW, FRAM consumes only 4.6 μW. According to [37], while the write voltage of EEPROM is 12 V, the write voltage of FRAM is 2 V. In addition, the lifetime of EEPROM in the sense of the number of possible write operation is limited to 10^6. On the other hand, the lifetime of FRAM in the sense of the number of possible write operation is 10^{12}.

In the future, when these memories will be used widely and implemented in tags, the power consumed by modules except the cryptographic module is predicted to be smaller. Thus, these memories will extend the maximum read distance when the lightweight cryptographic algorithm is used.

7.4.2 Usability of FPGA

To discuss the practicality of FPGA on the tag implementation, we compare FPGA and ASIC implementation results of the same circuits. From our ASIC implementation results, it has been cleared that there is a linear relationship between the power consumption of the circuit and the number of signal toggles. To evaluate the number of signal toggles, we have simulated the circuits' behavior for 100 random test vectors and have generated VCD files. We have calculated a population mean and a population standard by 99 % reliable section, and then we have predicted the number of signal toggles by three sigma limit. From these experimental results, the power consumption of the circuit can be evaluated by a liner equation.

$$Power = 0.2848 \cdot x, \tag{7.1}$$

where, x is the number of signal toggles. Order of the power is μW.

The circuits on FPGA consume power of mW order operating at 24 MHz [59]. It is predicted that the circuits on FPGA still consume power of mW order operating at 100 kHz since there is a linear relationship between the power and the clock frequency. Therefore, it seems difficult to use FPGA on the tag at the moment. When the power consumption of FPGA will be low in the future, FPGA will be integrated in a tag.

On the other hand, according to the results in Chapter 5, tendency of the evaluation results on FPGA are similar to the evaluation results on ASIC. Therefore, FPGA is useful to compare hardware performance of the cryptographic algorithms roughly and relatively.

7.4.3 Impact of Cryptographic Algorithm

The power consumption of the cryptographic block using SPONGENT-160 and Keccak are 262 μW and 534 μW, respectively. The cryptographic block using Keccak consumes two times

larger power compared with the one using SPONGENT-160. According to the equation (3.44), the maximum read distance of the tag inversely proportional to the square root of the power consumption. Therefore, the maximum read distance of the tag using Keccak is predicted to be 0.875 times shorter than that of the tag using SPONGENT-160. From the evaluation results of the tag, the maximum read distance of the tag using SPONGENT-160 and Keccak are 10 cm and 8.5 cm, respectively. Thus, we deduce that there is not much difference in the sense of the maximum read distance of the tag.

We have designed a passive tag with the analog power block that has enough capacity to supply power to the circuit absolutely. When we use smaller analog power block, the impact of the cryptographic block in the sense of power consumption will become larger. Then the lightweight cryptographic algorithm will have advantage.

7.5 Conclusion

In this chapter, we design, implement, and evaluate performance of a fully integrated passive UHF RFID tag that runs a privacy preserving authentication protocol based on hash functions. We design a single chip including the RF front end and the digital processing block. We choose SPONGENT-160 and Keccak. We implement all modules that are needed to run the whole authentication process including SRAM and EEPROM. Our evaluation results show that the maximum read distance is 10 cm when SPONGENT-160 is used. The time required for the first response of the tag including a hash calculation is less than 5 ms and the authentication is completed within 30 ms. The area requirements of the cryptographic block are 10 kGE. The average power consumption of the cryptographic block is 260 μW when the hash function is active.

When Keccak is used, the maximum read distance is 8.5 cm. The time required for the first response of the tag including a hash calculation is less than 0.3 ms and the authentication is completed within 20 ms. The area requirements of the cryptographic block are about 39.4 kGE. The average power consumption of the cryptographic block is 536 μW when the hash function is active. From our evaluation results, the RFID mutual authentication protocol based on a hash function is feasible for the passive UHF RFID tag.

Part VI

Conclusion

Chapter 8

Concluding Remarks and Future Research Direction

Conclusion

An RFID tag can be a key device in IoT system to collect physical information. The tag will be attached to items or owned by people. Since the tag returns a response whenever the tag receives a query from a reader, there are security and privacy issues on the tag. To overcome these issues, tag authentication protocols have been developed. The goal of this thesis has been to construct a design methodology of a secure RFID tag implementation. Specifically, we aim to make it difficult for an attacker to trace a tag by designing, implementing, and evaluating the tag that can run the authentication protocol based on a cryptographic algorithm.

In Part III, we have discussed hardware architecture and cryptographic algorithms for a tag. Concretely, we have focused on hash functions and stream ciphers, which are widely used for tag authentication protocols, and we have evaluated hardware performance. Operating time, area requirements, and power consumption are important evaluation metrics for a tag. Since the operating time needed for cryptographic process largely depends on hardware architecture, we have studied hardware architecture for the tag from a viewpoint of the operating time. Our results show that the parallel architecture is suitable for the hash functions. Both the parallel architecture and the sequential architecture are suitable for the stream ciphers. Then, we have implemented

the hash functions and the stream ciphers using the above mentioned hardware architecture, and we have evaluated hardware performance. From our evaluation results, when low-area requirements and low-power consumption are required, SPONGENT-160 is a good candidate. When high speed performance is required, Keccak can be used for a tag under the condition that the power consumption of the algorithm is allowed. In conclusion of Part III, the lightweight cryptographic algorithm contributes for a tag when low-area requirements and low-power consumption are required while it takes time for the cryptographic process. On the contrary, the standard cryptographic algorithm contributes for a tag when high speed performance is required while it takes much area requirements and consumes power.

In Part IV, we have discussed a lightweight pseudorandom number generator for a tag. In order to use a cryptographic circuit on the tag effectively, the cryptographic circuit should be used to generate pseudorandom number. Hence, we have proposed a lightweight pseudorandom number generator using a hash function. Our method is based on the hash chain, and the output of the hash function is divided into the updated seed and the pseudorandom number. Our evaluation results show that the pseudorandom number generated by our method passes NIST test suite and Diehard test suite. In conclusion of Part IV, the lightweight pseudorandom number generator we have proposed contributes for the security enhancement of the tag.

In Part V, we have designed, implemented, and evaluated a tag that runs the authentication protocol based on a cryptographic algorithm. We have designed a single chip including the analog front end block and the digital processing block. As the cryptographic algorithm, we have used SPONEGNT-160 and Keccak. We have done a silicon proof of the chip, and we have attached the antenna to the chip. Our evaluation results show that the tag has operated correctly. When SPONGENT-160 is activated, the maximum read distance of the tag, the power consumption, and the operating time for the authentication process are 10 cm, 260μW, and 30 ms, respectively. When Keccak is activated, the maximum read distance of the tag, the power consumption, and the operating time for the authentication process are 8.5 cm, 534μW, and 20 ms, respectively. In conclusion of Part V, the feasibility of the privacy preserving authentication protocol based on cryptographic algorithms is demonstrated. In addition, it is verified that the standard cryptographic algorithm works on the tag.

Future work

Future research direction should include developing a secure RFID tag that has resistance against side channel attacks. In fact, even if the tag runs the authentication protocol, which updates a secret key every time when the tag is authenticated, the secret key can be revealed. If immediately detach the tag from the reader after the tag finishes the cryptographic process and sends the response to the reader, the secret key is not updated. This is because the tag cannot receive power from the reader to rewrite the secret key in the memory. Therefore, the secret key is used to generate tag's response repeatedly. Thus, the secret key in the tag can be revealed by side channel attack such as differential electromagnetic analysis.

References

[1] G. Avoine, M.A. Bingöl, X. Carpent, and S. Kardas, "Deploying OSK on Low-Resource Mobile Devices," In *Security and Privacy Issues 9th International Workshop –RFIDsec 2013*, Vol. 8262 of Lecture Notes in Computer Science, Springer-Verlag, pp. 3-18, 2013.

[2] G. Avoine, M.A. Bingöl, X. Carpent, and S.B.O. Yalcin, "Privacy-Friendly Authentication in RFID Systems: On Sublinear Protocols Based on Symmetric-Key Cryptography," IEEE Trans. Mobile Computing, Vol. 12, No. 10, pp. 2037-2049, 2013.

[3] J.P. Aumasson, L. Henzen, W. Meier, and M. N-Plasencia, "QUARK: A Lightweight Hash," In *Workshop on Cryptographic Hardware and Embedded Systems –CHES 2010*, Vol. 6225 of Lecture Notes in Computer Science, Springer-Verlag, pp. 1-15, 2010.

[4] J.-P. Aumasson, L. Henzen, W. Meier,and R.C.-W. Phan, "SHA-3 proposal BLAKE," available at `http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/submissions_rnd3.html`, 2011.

[5] A. Aysu, E. Gulcan, D. Moriyama, P. Schaumont, and M. Yung, "End-To-End Design of a PUF-Based Privacy Preserving Authentication Protocol," In *Workshop on Cryptographic Hardware and Embedded Systems –CHES 2015*, Vol. 9293 of Lecture Notes in Computer Science, Springer-Verlag, pp. 556-576, 2015.

[6] J.P. Aumasson, L. Henzen, W. Meier, and M. N-Plasencia, "QUARK: A Lightweight Hash," Journal of Cryptology, Vol. 26, No. 2, pp. 313-339, 2013.

[7] M. Agren, M. Hell, T. Johansson, and W. Meier, "Grain-128a: a new version of Grain-128 with optional authentication," International Journal of Wireless and Mobile Computing, Vol. 5, No. 1, pp. 48-59, 2011.

[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 the proceedings of SAC 2000, Vol. 2012 of Lecture Notes in Computer Science, Springer-Verlag, pp. 39-56, 2001.

[9] Atmel Innovative Silicon RFID IDIC Solutions, 2013.

[10] D. J. Bernstein, "CubeHash specification," available at http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/submissions_rnd2.html, 2009.

[11] G.T. Becker, "The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs," In *Workshop on Cryptographic Hardware and Embedded Systems –CHES 2015*, Vol. 9293 of Lecture Notes in Computer Science, Springer-Verlag, pp. 535-555, 2015.

[12] C. Berbain, O. Billet, A. Canteaut, N. Courtois, B. Debraize, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert, "DECIM v2," In New Stream Cipher Designs, Vol. 4986 of Lecture Notes in Computer Science, Springer-Verlag, pp. 140-151, 2008.

[13] R. Benadjila, O. Billet, H. Gilbert, G. Macario-Rat, T. Peyrin, M. Robshaw, and Y. Seurin, "SHA-3 Proposal: ECHO," available at http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/submissions_rnd2.html, 2009.

[14] E. Bresson, A. Canteaut, B.C-Mames, C. Clavier, T. Fuhr, A. Gouget, T. Icart, J.-F. Misarsky, M. Naya-Plasencia, P. Paillier, T. Pornin, J.-R. Reinhard, C. Thuillet, and M. Videau, "Shabal, A Submission to NIST's Cryptographic Hash Algorithm Competition," available at http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/submissions_rnd2.html, 2009.

[15] E. Biham and O. Dunkelman, "The SHAvite-3 Hash Function," available at http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/submissions_rnd2.html, 2009.

[16] L. Batina, A. Das, B. Ege, E.B. Kavun, N. Mentens, C. Paar, I. Verbauwhede, and T. Yalcin, "Dietary Recommendations for Lightweight Block Ciphers: Power, Energy and Area Analysis of Recently Developed Architectures," In *Security and Privacy Issues 9th International Workshop –RFIDsec 2013*, Vol. 8262 of Lecture Notes in Computer Science, Springer-Verlag, pp. 103-112, 2013.

[17] G. Bertoni, J. Daemen, M. Peeters, and G.V. Assche, "Cryptographic sponge functions (2011)," available at http://sponge.noekeon.org/CSF-0.1.pdf

[18] G. Bertoni, J. Daemen, M. Peeters, and G.V. Assche, "The Making of KECCAK," Cryptologia, Vol. 38, No. 1, pp. 26-60, 2014.

[19] A. Bogdanov, M. Knežević, G. Leander, D. Toz, K. Varıcı, and I. Verbauwhede, "SPONGENT: A Lightweight Hash Function," In *Workshop on Cryptographic Hardware and Embedded Systems –CHES 2011*, Vol. 6917 of Lecture Notes in Computer Science, Springer-Verlag, pp. 312-325, 2011.

[20] A. Bogdanov, M. Knežević, G. Leander, D. Toz, K. Varıcı, and I. Verbauwhede, "SPONGENT: The Design Space of Lightweight Cryptographic Hashing," IEEE Trans. Computers, Vol. 62, No. 10, pp. 2041-2053, 2013.

[21] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An Ultra-Lightweight Block Cipher," In *Workshop on Cryptographic Hardware and Embedded Systems –CHES 2007*, Vol. 4727 of Lecture Notes in Computer Science, Springer-Verlag, pp. 450-466, 2007.

[22] P. Bulens, F.-X. Standaert, J.-J. Quisquater, P. Pellegrin, and G. Rouvroy, "Implementation of the AES-128 on Virtex-5 FPGAs," in *Progress in Cryptology –AFRICACRYPT 2008*, Vol. 5023 of Lecture Notes in Computer Science, Springer-Verlag, pp.16-26, 2008.

[23] M. Boesgaard, M. Vesterager, T. Pedersen, J. Christiansen, and O. Scavenius, " Rabbit: A new high-performance stream cipher," In *10th International Workshop –FSE 2003*, Vol.2887 of of the series Lecture Notes in Computer Science, Springer-Verlag, pp. 307-329, 2003.

[24] C. Cakir, M. Bhargava, and K. Mai,"6T SRAM and 3T DRAM Data Retention and Remanence Characterization in 65nm bulk CMOS," In *Proceedings of the IEEE 2012 Custom Integrated Circuits Conference*, IEEE, pp. 1-4, 2012.

[25] C.D. Cannière and B. Preneel, "TRIVIUM," In New Stream Cipher Designs, Vol. 4986 of Lecture Notes in Computer Science, Springer-Verlag, pp. 244-266, 2008.

[26] C.D. Cannière and B. Preneel, "TRIVIUM Specifications", available at http://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf.

[27] J. Chen, A. Miyaji, H. Sato, and C. Su, " Improved Lightweight Pseudo-Random Number Generators for the Low-Cost RFID Tags," IEEE Trustcom/BigDataSE/ISPA , Vol. 1, pp. 17-24, 2015.

[28] A.P. Chandrakasan, S. Sheng, and R.W. Broodersen, "Low Power Digital CMOS Design," IEEE Journal of Solid-State Circuits, Vol. 27, No.4, pp. 473-484, 1992.

[29] C.D. Cannière, H. Sato, and D. Watanabe, "Hash Function Luffa: Specification," available at http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/submissions_rnd2.html, 2009.

[30] "The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness," available at http://stat.fsu.edu/pub/diehard/.

[31] V. Dixit, H.K. Verma, and A.K. Singh, "Comparison of various Security Protocols in RFID," International Journal of Computer Applications, Vol. 24, No. 7, pp. 17-21, 2011.

[32] P. Ekdahl and T. Jhansson, "A New Version of the Stream Cipher SNOW," In *9th Annual International Workshop –SAC 2002*, Vol.2595 of of the series Lecture Notes in Computer Science, Springer-Verlag, pp. 47-61, 2002.

[33] The eSTREAM Project, available at http://www.ecrypt.eu.org/stream/.

[34] EPCglobal Inc., "EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz - 960 MHz Version 1.2.0, 2008," available at http://www.gs1.org/gsmp/kc/epcglobal/uhfc1g2/uhfc1g2_1_2_0-standard-20080511.pdf.

[35] GS1 AISBL, "GS1 EPC Tag Data Standard 1.6, 2011," available at http://www.gs1.org/gsmp/kc/epcglobal/tds/tds_1_6-RatifiedStd-20110922.pdf.

[36] J. Ertl, T. Plos, M. Feldhofer, N. Felber, and L. Henzen, "A Security-Enhanced UHF RFID Tag Chip," In *2013 Euromicro Conference on Digital System Design*, IEEE, pp. 705-712, 2013.

[37] K. Finkenzeller, "RFID Handbook" .

[38] FUJITSU SEMICONDUCTOR, "MEMORY MANUAL," 2010.

[39] H.T. Friis, "A Note on a Simple Transmission Formula," In Proceedings of the IRE, IEEE, Vol. 34, No. 5, pp. 254-256, 1946.

[40] N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno, J. Callas, and J. Walker, "The Skein Hash Function Family," available at http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/submissions_rnd3.html, 2011.

[41] M. Feldhofer and J. Wolkerstorfer, "Strong Crypto for RFID Tags - A Comparison of Low-Power Hardware Implementations," In *ISCAS 2007*, pp. 1839-1842, 2007.

[42] F. Gray, "Pulse Code Communication," 1953.

[43] T. Good and M. Benaissa, "ASIC Hardware Performance," In New Stream Cipher Designs, Vol. 4986 of Lecture Notes in Computer Science, Springer-Verlag, pp. 267-293, 2008.

[44] F.D. Garcia, G.D.K. Gans, R. Muijrers, P.V. Rossum, R. Verdult, R.W. Schreur, and B. Jacobs, "Dismantling MIFARE Classic," In *13th Europian Symposium on Research in Computer Security, ESORICS 2008*, 2008.

[45] G. Gans, J-H. Hoepman, and F.D. Garcia, "A Practical Attack on the MIFARE Classic," arXix:0803.2285v2, [cs.CR] 2008.

[46] D. Gligoroski, V. Klima, S.J. Knapskog, M. El-Hadedy, J. Amundsen, and S.F. Mjolsnes, "Cryptographic Hash Function BLUE MIDNIGHT WISH," available at http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/submissions_rnd2.html, 2009.

[47] P. Gauravaram, L.R. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schläffe,and S. Thomsen, "Grøstl A SHA-3 Candidate," available at http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/submissions_rnd3.html, 2011.

[48] J. Guo, T. Peyrin, and A. Poschmann, "The PHOTON Family of Lightweight Hash Functions," In *Advances in Cryptology –CRYPTO 2011*, Vol. 6841 of Lecture Notes in Computer Science, Springer-Verlag, pp. 222-239, 2011.

[49] F.D. Garcia, P.V. Rossum, R. Verdult, and R.W. Schreur, "Wirelessly Pickpocketing a Mifare Classic Card," IEEE Symposium on Security and Privacy, 2009.

[50] Hitachi, Ltd., "Pseudorandom number generator Enocoro," available at http://www.hitachi.co.jp/rd/yrl/crypto/enocoro/index.html.

[51] GSI Information Security Group, "RFID Security & Privacy Rounge," available at http://www.avoine.net/rfid/.

[52] S. Halevi, W.E. Hall, and C.S. Jutla, "The Hash Function Fugue," available at http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/submissions_rnd2.html, 2009.

[53] M. Hell, T. Johansson, A. Maximov, and W. Meier, "The Grain Family of Stream Ciphers," In New Stream Cipher Designs, Vol. 4986 of Lecture Notes in Computer Science, Springer-Verlag, pp. 179-190, 2008.

[54] M. Hell, T. Johansson, and W. Meier, "Grain: a stream cipher for constrained environments," International Journal of Wireless and Mobile Computing, Vol. 2, No. 1, pp. 86-93, 2007.

[55] Y. Hanatani, M. Ohkubo, S. Matsuo, K. Sakiyama, and K. Ohta, "A Study on Computational Formal Verification for Practical Cryptographic Protocol: The Case of Synchronous RFID Authentication," In *Financial Cryptography and Data Security Workshops –FC 2011*, Vol. 7126 of Lecture Notes in Computer Science, Springer-Verlag, pp. 70-87, 2012.

[56] IDTechEx Ltd. "RFID Forecasts, Players and Opportunities 2016-2026," 2015.

[57] *Ö*. K*üçü*k, "The Hash Function Hamsi," available at http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/submissions_rnd2.html, 2009.

[58] S. Kerckhof, F. Durvaux, C. Hocquet, D. Bol, and F.X. Standaert, "Towards Green Cryptography: A Comparison of Lightweight Ciphers from the Energy Viewpoint," In *Workshop on Cryptographic Hardware and Embedded Systems –CHES 2012*, Vol. 7428 of Lecture Notes in Computer Science, Springer-Verlag, pp. 390-407, 2012.

[59] M. Knežević, K. Kobayashi, J. Ikegami, S. Matsuo, A. Satoh, U. Kocabas, J. Fan, T. Katashita, T. Sugawara, K. Sakiyama, I. Verbauwhede, K. Ohta, N. Homma, and T. Aoki, "Fair and Consistent Hardware Evaluation of Fourteen Round Two SHA-3 Candidates," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 20, No. 5, pp. 827-840, 2012.

[60] R.S. Katti and S.K. Srinivasan, "Efficient hardware implementation of a new pseudo-random bit sequence generator," In *IEEE International Symposium on Circuits and Systems*, pp. 1393-1396, 2009.

[61] S. Kiyomoto, T. Tanaka, and K. Sakurai, "K2 Stream Cipher," In *4th International Conference –ICETE 2007*, Vol. 23 of the series Communications in Computer and Information Science, Springer-Verlag, pp. 214-226, 2007.

[62] E.B. Kavun and T. Yalcin, "A Lightweight Implementation of Keccak Hash Function for Radio-Frequency Identification Applications," In *6th International Workshop –RFIDsec 2010*, Vol. 6370 of Lecture Notes in Computer Science, Springer-Verlag, pp. 258-269, 2010.

[63] G. Leurent, C. Bouillaguet, and P.-A. Fouque, "SIMD is a Message Digest," available at `http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/submissions_rnd2.html`, 2009.

[64] Z. Liu, D. Liu, L. Li, H. Lin, and Z. Yong, "Implementation of a New RFID Authentication Protocol for EPC Gen2 Standard," Sensors Journal, IEEE, Vol. 15, No. 2, pp. 1003-1011, 2015.

[65] P. P-Lopez, E.S. Millan, J.C.A. Lubbe, and L.A. Entrena, "Cryptographically secure pseudo-random bit generator for RFID tags," In *IEEE International Conference for Internet Technology and Secured Transactions (ICITST)*, pp. 1-6, 2010.

[66] Y. Li, S. Mikami, D. Watanabe, K. Ohta, and K. Sakiyama, "Single-Chip Implementation and Evaluation of Passive UHF RFID Tag with Hash-Based Mutual Authentication," In *Workshop on RFID Security, RFIDsec'14 Asia*, Cryptology and Information Security Series, IOS Press, Vol. 12, pp. 3-15, 2014.

[67] Libelium Univeils the Top50 Internet of Things Applications, available at `http://www.itwire.com/opinion-and-analysis/beerfiles/54432-libelium-unveils-the-top-50-internet-of-things-applications`.

[68] M. Merhi, J.C. H-Castro, and P. P-Lopez, "Studying the Pseudo Random Number Generator of a low-cost RFID tag," In *IEEE International Conference on RFID-Technologies and Applications*, 2011.

[69] U. Mujahid, M. Najam-ul-silam, A. Sharif, T. Khan, A. Qavi, and Bilal, "A novel lightweight pseudorandom number generator for passive RFID systems," In *IEEE 17th International Multi-Topic Conference (INMIC)*, pp. 149-154, 2014.

[70] S. Mangard, E. Oswald, and T. Popp, "Power Analysis Attacks Revealing the Secrets of Smart Cards," Springer, 2007.

[71] H. Martin, E.S. Millan, L. Entrena, P.P. Lopez, and J.C.H. Castro, "AKARI-X: A pseudorandom number generator for secure lightweight systems," In *IEEE 17th International On-Line Testing Symposium*, pp. 228-233, 2011.

[72] H. Martin, E. San Millan, P. Peris-Lopez, and J.E. Tapiador, "Efficient ASIC Implementation and Analysis of Two EPC-C1G2 RFID Authentication Protocols," Sensors Journal, IEEE, Vol. 13, No. 10, pp. 3537-3547, 2013.

[73] A.J. Menezes, P.C.van Oorschot, and S.A. Vanstone, "Handbook of APPLIED CRYPTOGRAPHY," 1996.

[74] A. Moradi, M. Salmasizadeh, and M.T.M. Shalmani, "Power Analysis Attacks on MDPL and DRSL Implementations," In *Information Security and Cryptology – ICISC 2007*, Vol. 4817 of Lecture Notes in Computer Science, Springer-Verlag, pp. 259–272, 2007.

[75] S. Mikami, H. Yoshida, D. Watanabe, and K. Sakiyama, "Correlation Power Analysis and Countermeasure on the Stream Cipher Enocoro-128v2," IEICE Trans. Fundam. Electron. Commun. Comput. Sci., Vol.96-A, No.3, pp. 697-704, 2013.

[76] S. Mikami, D. Watanabe, and K. Sakiyama, "A Comparative Study of Stream Ciphers and Hash Functions for RFID Authentication Protocol," In *The 2013 Workshop on RFID and IoT Security (RFIDsec'13 Asia)*, Cryptology and Information Security Series, IOS Press, Vol. 11, pp. 83-94, 2013.

[77] S. Mikami, D. Watanabe, and K. Sakiyama, "A Performance Evaluation of Cryptographic Algorithms on FPGA and ASIC on RFID Design Flow," In *International Conference on Information and Communication Technology (ICoICT'16)*, IEEE, (May, 2016).

[78] K. Nohl, D. Evans, and H. Plotz, "Reverse-Engineering a Cryptographic RFID Tag," USENIX Security Symposium, 2008.

[79] National Institute for Standards and Technology, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators," NIST Special Publication 800-90A, January 2012, available at `http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf`.

[80] National Institute for Standards and Technology, “Specification for the ADVANCED ENCRYPTION STANDARD (AES),” NIST Federal Information Processing Standards Publication 197, 2001, available at http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[81] NIST Special Publication 800-22 Revision 1a, “A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications,” 2010, available at http://csrc.nist.gov/publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf.

[82] National Institute for Standards and Technology, “Recommendation for Key Management: Part 1: General (Revision 4),” NIST Special Publication 800-57, 2016, available at http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf.

[83] National Institute for Standards and Technology, “Secure Hash Standard (SHS),” NIST Federal Information Processing Standards Publication 180-4, 2015, available at http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf.

[84] National Institute for Standards and Technology, “Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher,” NIST Special Publication 800-67, 2012, available at http://csrc.nist.gov/publications/nistpubs/800-67-Rev1/SP-800-67-Rev1.pdf.

[85] S. Nikova, C. Rechberger, and V. Rijmen, “Threshold Implementations Against Side-Channel Attacks and Glitches,” In *8th International Conference on Information and Communications Security –ICICS 2006*, Vol.4307 of of the series Lecture Notes in Computer Science, Springer-Verlag, pp. 529-545, 2006.

[86] NXP Semiconductors, “MIFARE DESFire EV1”.

[87] ORIDAO, “Secured EPC Gen2 RFID ASIC”.

[88] M. Ohkubo, K. Suzuki, and S. Kinoshita, “RFID privacy issues and technical challenges,” Commun. ACM, Vol. 48, No. 9, pp. 66-71, 2005.

[89] M. Ohkubo, K. Suzuki, and K. Kinoshita, “Cryptographic Approach to ”Privacy-Friendly” Tags,” RFID Privacy Workshop. MIT, MA 2003.

[90] S. Piramuthu, "Lightweight Cryptographic Authentication in Passive RFID-Tagged Systems," IEEE Trans. Systems, man, and cybernetics -part C Applications and Reviews, Vol. 38, No. 3, pp. 360-376, 2008.

[91] A. Poschmann, M.J.B. Robshaw, F. Vater, and C. Paar, "Lightweight Cryptography and RFID: Tackling the Hidden Overhead," KSII Trans. Internet and Information Systems, Vol. 4, No. 2, pp. 98-116, 2010.

[92] M. Qazi, M. Clinton, S. Bartling, and A.P. Chandrakasan, "A Low-Voltage 1 Mb FRAM in 0.13 μm CMOS Featuring Time-to-Digital Sensing for Expanded Operating Margin," IEEE Journal of Solid-State Circuits, Vol. 47, No. 1, pp. 141-150, 2012.

[93] RAMTRON, "Performance and Energy Comparison of F-RAM and EEPROM Devices," available at `http://www.digikey.ca/Web\%20Export/Supplier\%20Content/ramtron-1140/pdf/ramtron-tech-fram-eeprom-comparison.pdf?redirected=1`.

[94] J. M-Segui, J. G-Alfaro, and J. H-Joancomarti, "Analysis and Improvement of a Pseudorandom Number Generator for EPC Gen2 Tags," Financial Cryptography and Data Security, Vol. 6054 of the series Lecture Notes in Computer Science, Springer, pp. 34-46, 2010.

[95] J. M-Segui, J. G-Alfaro, and J. H-Joancomarti, "Multiple-polynomial LFSR based pseudorandom number generator for EPC Gen2 RFID tags," In *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*, pp. 3820-3825, 2011.

[96] M.J.O. Saarinen and D. Engels, "A Do-It-All-Cipher for RFID: Design Requirements (Extended Abstract)," available at `http://eprint.iacr.org/2012/317.pdf`.

[97] T. Shirai, K. Shibutani, T. Akishita, S. Moriai, and T. Iwata, "The 128-Bit Blockcipher CLEFIA," In *14th International Workshop –FSE 2007*, Vol.4593 of the series Lecture Notes in Computer Science, Springer-Verlag, pp. 181-195, 2007.

[98] D. Suzuki, M. Saeki, and T. Ichikawa, "DPA Leakage Models for CMOS Logic Circuits," In *Workshop on Cryptographic Hardware and Embedded Systems –CHES 2005*, Vol. 3659 of Lecture Notes in Computer Science, Springer-Verlag, pp. 366-382, 2005.

[99] D.-Z. Sun and J.-D. Zhong, "A Hash-Based RFID Security Protocol for Strong Privacy

Protection," IEEE Trans. Consumer Electronics, Vol. 58, No. 4, pp. 1246-1252, 2012.

[100] M. Teramura, M. Shiozaki, T. Okamoto, T. Murayama, and T. Fujino, "Optimization Method of RG-DTM PUF for Authentication and Random Number Generation," Symposium on Cryptography and Information Security 2013.

[101] Available at `http://www.tsensorssummit.org/`.

[102] K. Tiri and I. Verbauwhede, "A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation," In *Design, Automation and Test in Europe Coneference - DATE 2004*, pp. 246-251, 2004.

[103] H. Wu, "The Hash Function JH," available at `http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/submissions_rnd3.html`, 2011.

[104] D. Watanabe, S. Furuya, H. Yoshida, K. Takaragi, and B. Preneel, "A New Keystream Generator MUGI," In *9th International Workshop, –FSE 2002*, Vol. 2565 of of the series Lecture Notes in Computer Science, Springer-Verlag, pp. 179-194, 2002.

[105] J. Wang, H. Li, and F. Yu, "Design of Secure and Low-cost RFID Tag Baseband," In *International Conference on Wireless Communications, Networking and Mobile Computing, WiCom 2007*, pp. 2066-2069, 2007.

Paper Reuse Permission

We obtained from IEEE, the permission to reuse the contents described in Chapter 5 which were published as a conference paper:

Shugo Mikami, Dai Watanabe, and Kazuo Sakiyama, "A Performance Evaluation of Cryptographic Algorithms on FPGA and ASIC on RFID Design Flow," In Proc. International Conference on Information and Communication Technology (ICoICT'16), IEEE, 2016.

Title:	A performance evaluation of cryptographic algorithms on FPGA and ASIC on RFID design flow
Conference Proceedings:	Information and Communication Technology (ICoICT), 2016 4th International Conference on
Author:	Shugo Mikami
Publisher:	IEEE
Date:	May 2016

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

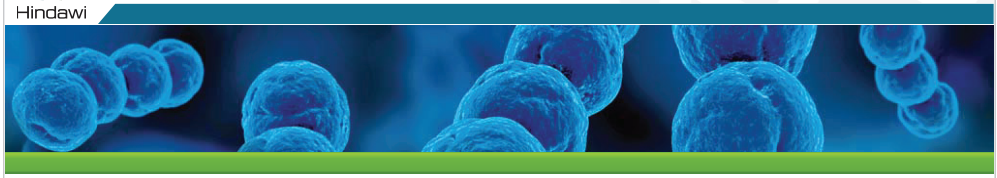
If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK ××× CLOSE WINDOW

The contents described in Chapter 7 was published as a journal paper:

Shugo Mikami, Dai Watanabe, Yang Li, and Kazuo Sakiyama, "Fully Integrated Passive UHF RFID Tag for Hash-Based Mutual Authentication Protocol," The Scientific World Journal, Hindawi, Volume 2015 (2015), Article ID 498610, 11 pages, Aug., 2015.

Hindawi Publishing Corporation



Information Menu
- Abstracting and Indexing
- Browse Journals
- Open Access Memberships
- Publication Ethics
- Resources and Tools
- Spotlight Articles
- Subscription Information

OPEN ACCESS

Login to the Manuscript Tracking System

Hindawi Copyright and License Agreement

By submitting your work, you signify that you have read and agreed to the following terms.

In consideration for the publication of your article by Hindawi you hereby agree to pay the agreed fee and grant to Hindawi an irrevocable nonexclusive license to publish in print and electronic format, and further sublicense the article, for the full legal term of copyright and any renewals thereof in all languages throughout the world in all formats, and through any medium of communication.

You shall retain the perpetual royalty-free right to reproduce and publish in print and electronic format, and further sublicense the article in all languages throughout the world in all formats, and through any medium of communication provided that you make reference to the first publication by the Journal and Hindawi.

Upon publication, your article shall be openly licensed using the Creative Commons Attribution 4.0 License (http://creativecommons.org/licenses/by/4.0/). In addition, any data related to your article, including its reference list(s) and its additional files, shall be distributed under the terms of the Creative Commons Public Domain Dedication waiver (http://creativecommons.org/publicdomain/zero/1.0/).

- You warrant that the article is your original work.
- You warrant that you are the sole author(s) of the article and have full authority to enter into this Agreement and in granting rights to Hindawi Publishing Corporation that are not in breach of any other obligation.
- You warrant that the article is submitted for first publication in the journal and that it is not being considered for publication elsewhere and has not already been published elsewhere, either in printed or electronic form.
- You warrant that you have obtained all necessary permissions for the reproduction of any copyright works (including artistic works, e.g., illustrations, photographs, charts, maps, and other visual material) contained in the article and not owned by you and you have acknowledged all the sources.
- You warrant that the article contains no violation of any existing copyright, other third party rights, or any libelous statements, and does not infringe any rights of others.
- You warrant that the Contribution contains no violation of any existing copyright, other third party rights, or any libellous statements, and does not infringe any rights of others.
- You warrant that you have taken due care to ensure the integrity of the article so that to your—and currently accepted scientific—knowledge all statements contained in it purporting to be facts are true and any formula or instruction contained in the article will not, if followed accurately, cause any injury, illness, or damage to the user.
- You warrant that the article, in the form to be delivered to Hindawi, does not contain information that may violate the State Secrets Laws of the People's Republic of China ("PRC"), including (a) any information that has a vital bearing on Chinese state security and national interests; (b) any information defined as a "State Secret" under the PRC law, or (c) any classified information belonging to governmental authorities of the People's Republic of China, including government agencies, quasi-government agencies, public institutions, or state-owned enterprises (together the "State Secret Laws"). For the purposes of PRC law and this Agreement "State Secrets" shall include, but not be limited to: (i) unpublished information concerning major policy decisions on Chinese State affairs; (ii) confidential information concerning national defense and the activities of the armed forces; (iii) confidential information concerning national diplomatic policies and activities; (iv) confidential information concerning national economic and social development; (v) matters concerning classified science and technology; and (vi) unpublished Chinese State security matters and non-public information about the on-going investigation of criminal offenses.

You agree to indemnify Hindawi Publishing Corporation against any claims in respect of the above warranties.

You are urged to screen out any information that may constitute State Secrets under the law of the PRC or violate the State Secret Laws before submitting your article to Hindawi. Hindawi is not responsible for any consequences from inadvertent disclosure of such State Secrets or violation of the State Secret Laws in your work, and Hindawi makes no warranty, expressed, or implied, with respect to the contents of the publications. Hindawi does not assume any liability for any infringement of the State Secret Laws arising from its publication.

While understanding that copyright remains your own as the author, you hereby authorize Hindawi to act on your behalf to defend your copyright should it be infringed, and to retain half of any damages awarded, after deducting costs.

No amendment or modification of any provision of this Agreement shall be valid or binding unless made in writing and signed by all parties. This Agreement constitutes the entire agreement between the parties with respect to its subject matter, and supersedes all previous agreements, understandings, and representations. The invalidity or unenforceability of any particular provision of this Agreement shall not affect the other provisions, and this Agreement shall be construed in all respects as if any invalid or unenforceable provision was omitted.

You agree that this Agreement shall be deemed to be a contract made in Egypt, and shall be construed and applied in all respects in accordance with Egyptian law. You submit to the jurisdiction of the Egyptian courts.

Acknowledgements

I would like to take this opportunity to thank the people who have helped and encouraged me during the past years.

First of all, I would like to express my deep gratitude to my supervisor, Professor Kazuo Sakiyama for giving me a lot of research opportunities, much useful advices in research discussions, and continuous encouragement during this work. Without his kind guidance, it would not have been possible for me to complete this thesis.

I am also grateful to all member of supervisory committee for my PhD defense - Professor Kazuo Ohta, Professor Hiroshi Yoshiura, Professor Koichiro Ishibashi, and Associate Professor Mitsugu Iwamoto - for their valuable advices and suggestion to improve the presentation.

I am thankful to Dr. Kazuo Takaragi and Professor Yasuko Fukuzawa for giving me the opportunity to work as a cryptographic researcher. Without their decision, I did not have opportunity to learn how the cryptography supports our daily life and did not work in the crypto area.

I have been very fortunate to work with my other co-authors Dr. Dai Watanabe, Dr. Yang Li, Dr. Hirotaka Yoshida, Mr. Nagamasa Mizushima, and Ms. Setsuko Nakamura. I thank for the interesting discussions.

I am very grateful for the Hitachi directors who have supported me including Dr. Masahiro Mimura and Dr. Tadashi Kaji. I would like to express gratitude to my colleagues in Hitachi including Dr. Makoto Kayashima, Mr. Toru Owada, Dr. Hiroki Uchiyama, Dr. Kota Ideguchi, Dr. Kunihiko Miyazaki, Dr. Masayuki Yoshino, and Mr. Ken Naganuma.

I really thank cryptographic researchers in the other organizations. I thank Professor Naofumi Homma, Professor Makoto Nagata, and Assistant Professor Keisuke Hakuta for your collaboration.

A special thank you goes to Ms. Yoko Ishii and Ms. Sachie Kasai for their kind support at the University of Electro-Communications (UEC). I would like to thank all the others belong to Sakiyama Laboratory in UEC.

I have special appreciation to Professor Nobuyuki Imoto, Professor Masato Koashi, and Associate Professor Takashi Yamamoto who were my supervisors when I was a graduate student at Osaka University. They provided me a foundation for a researcher. I thank my colleagues Dr. Koji Azuma and the othres belongs to Imoto Laboratory for help.

I am very grateful to my family for their constant support and encouragement. Without their help, I could not have done this work.

Shugo Mikami

March 2017

List of Publications Related and Referred to the Dissertation

Related Publications

Journal Papers

1. Shugo Mikami, Dai Watanabe, Yang Li, and Kazuo Sakiyama, "Fully Integrated Passive UHF RFID Tag for Hash-Based Mutual Authentication Protocol," The Scientific World Journal, Hindawi, Volume 2015 (2015), Article ID 498610, 11 pages, Aug., 2015.

Referred Conference Papers (with Formal Proceedings)

1. Shugo Mikami, Dai Watanabe, and Kazuo Sakiyama, "A Performance Evaluation of Cryptographic Algorithms on FPGA and ASIC on RFID Design Flow," In Proc. International Conference on Information and Communication Technology (ICoICT'16), IEEE, 2016.

Referred Publications

Journal Papers

1. Shugo Mikami, Hirotaka Yoshida, Dai Watanabe, and Kazuo Sakiyama, "Correlation Power Analysis and Countermeasure on the Stream Cipher Enocoro-128v2," IEICE Trans.

Fundam. Electron. Commun. Comput. Sci., Vol.96-A, No.3, pp. 697-704, 2013.

Referred Conference Papers (with Formal Proceedings)

1. Shugo Mikami, Dai Watanabe, and Kazuo Sakiyama, “A Comparative Study of Stream Ciphers and Hash Functions for RFID Authentications,” In Proc. The 2013 Workshop on RFID and IoT Security (RFIDsec’13 Asia), IOS Press, pp.83-94, 2013.

List of All Publications

Journal Papers

1. Shugo Mikami, Dai Watanabe, Yang Li, and Kazuo Sakiyama, “Fully Integrated Passive UHF RFID Tag for Hash-Based Mutual Authentication Protocol,” The Scientific World Journal, Hindawi, Volume 2015 (2015), Article ID 498610, 11 pages, Aug., 2015.
2. Shugo Mikami, Hirotaka Yoshida, Dai Watanabe, and Kazuo Sakiyama, “Correlation Power Analysis and Countermeasure on the Stream Cipher Enocoro-128v2,” IEICE Trans. Fundam. Electron. Commun. Comput. Sci., Vol.96-A, No.3, pp.697-704, 2013.
3. 三上　修吾，渡辺　大，水島　永雅，中村　節子，“ハッシュ関数 Luffa v2 の軽量ハードウェア実装及び評価，” 電子情報通信学会論文誌 A，Vol.J95-A，No.5，pp.407-415，2012.

Refereed Conference Papers (with Formal Proceedings)

1. Shugo Mikami, Dai Watanabe, and Kazuo Sakiyama, “A Performance Evaluation of Cryptographic Algorithms on FPGA and ASIC on RFID Design Flow,” In Proc. International Conference on Information and Communication Technology (ICoICT’16), IEEE, 2016.
2. Yang Li, Shugo Mikami, Dai Watanabe, Kazuo Ohta, and Kazuo Sakiyama, “Single-Chip Implementation and Evaluation of Passive UHF RFID Tag with Hash-Based Mutual Authentication,” In Proc. Workshop on RFID Security (RFIDsec’14 Asia), IOS Press, pp.3-15, 2014.
3. Shugo Mikami, Dai Watanabe, and Kazuo Sakiyama, “A Comparative Study of Stream

Ciphers and Hash Functions for RFID Authentications," In Proc. The 2013 Workshop on RFID and IoT Security (RFIDsec'13 Asia), IOS Press, pp.83-94, 2013.

Presentations

1. 三上　修吾, 渡辺　大, 崎山　一男, "バッファを用いた軽量擬似乱数生成器のグリッチ削減方法とハードウェア実装評価," コンピュータセキュリティシンポジウム 2014（CSS2014）, 1C3, 2014.
2. 三上　修吾，渡辺　大，水島　永雅，中村　節子，"ハッシュ関数 Luffa のハードウェア実装及び評価，" コンピュータセキュリティシンポジウム 2010（CSS2010），1B2，2010.
3. 三上　修吾，渡辺　大，"ストリーム暗号 Enocoro-128 v2 に対する CPA," 電子情報通信学会技術研究報告　信学技報 Vol.111, No.337, ISEC2011-62, pp.1-6, 2011.
4. 三上　修吾，吉田　博隆，渡辺　大，崎山　一男, "Threshold Implementation を利用したストリーム暗号 Enocoro-128 v2 の相関電力解析対策，" 2012 年暗号と情報セキュリティシンポジウム (SCIS2012)，2C2-1, 2012.
5. 三上　修吾，渡辺　大，"ストリーム暗号 Enocoro-128 v2 のソフトウェアおよびハードウェア実装と評価，" コンピュータセキュリティシンポジウム 2012（CSS2012），3C1，2012.
6. 三上　修吾，渡辺　大，崎山　一男，"RFID 認証プロトコル向け軽量暗号プロトコルの実装評価，" 2013 年暗号と情報セキュリティシンポジウム (SCIS2013)，2E2-1，2013.
7. 三上　修吾，渡辺　大，崎山　一男，"バッファを用いた軽量擬似乱数生成器のハードウェア実装と評価，" 2014 年暗号と情報セキュリティシンポジウム (SCIS2014), 2A2-1, 2014.
8. 李　陽, 三上　修吾，渡辺　大，太田　和夫，崎山　一男，"RFID システムにおけるリレー攻撃対策，" Hot Channel Workshop 2014.
9. 三上　修吾，渡辺　大，崎山　一男，"バッファを用いた軽量擬似乱数生成器のグリッチ削減方法と実装評価，" Hot Channel Workshop 2014.
10. 三上　修吾，渡辺　大，崎山　一男，"OSK プロトコル向け軽量暗号アルゴリズムの実

装評価，” Hot Channel Workshop 2013.

11. 三上　修吾，渡辺　大，“プライバシ保護を考慮した RFID 向けセキュリティ要件と軽量暗号アルゴリズムに関する一考察，” Hot Channel Workshop 2012.
12. Shugo Mikami, Dai Watanabe, Yang Li, and Kazuo Sakiyama, “Design Methods for Secure Hardware,” in NII Shonan Meeting, 2014.

Author Biography

Shugo Mikami was born on 28th April in Fukui, Japan. He received the B.E. and M.E. degrees from Osaka University, Osaka in Japan, in March 2006 and March 2008, respectively. He joined systems development laboratory Hitachi, Ltd in Japan. He has worked on the research and development on security. His research area includes hardware implementation of cryptographic algorithm, side channel attack against cryptographic circuit and countermeasure technique. Since April 2014, he has been working towards a PhD under the supervision of Professor Kazuo Sakiyama, at the Department of Informatics in the Graduate School of Informatics and Engineering, The University of Electro-Communications, Tokyo in Japan. Since April 2016, he is a member of study group ITU-T SG17 (Security). He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE) of Japan.