석사 학위논문
Master's Thesis

Sponge 함수에 기반한 인증암호화 기법 설계 및 분석

Design and Analysis of Authenticated Encryption Scheme based on the Sponge Function

김 학 주 (金 學 柱 Kim, HakJu)
전산학과
School of Computing

KAIST

2015

Sponge 함수에 기반한 인증암호화 기법 설계 및 분석

Design and Analysis of Authenticated Encryption Scheme based on the Sponge Function

Design and Analysis of Authenticated Encryption Scheme based on the Sponge Function

Advisor : Professor Kim, Kwangjo

by

Kim, HakJu
School of Computing
KAIST

A thesis submitted to the faculty of KAIST in partial fulfillment of the requirements for the degree of Master of Science in Engineering in the School of Computing . The study was conducted in accordance with Code of Research Ethics[1].

2015. 6. 11.

Approved by

Professor Kim, Kwangjo

[Advisor]

[1]Declaration of Ethical Conduct in Research: I, as a graduate student of KAIST, hereby declare that I have not committed any acts that may damage the credibility of my research. These include, but are not limited to: falsification, thesis written by someone else, distortion of research findings or plagiarism. I affirm that my thesis contains honest conclusions based on my own careful research under the guidance of my thesis advisor.

Sponge 함수에 기반한 인증암호화 기법 설계 및 분석

김 학 주

위 논문은 한국과학기술원 석사학위논문으로
학위논문심사위원회에서 심사 통과하였음.

2015년 6월 11일

심사위원장 김 광 조 (인)

심사위원 김 대 영 (인)

심사위원 김 순 태 (인)

MCS
20134452

김 학 주. Kim, HakJu. Design and Analysis of Authenticated Encryption Scheme based on the Sponge Function. Sponge 함수에 기반한 인증암호화 기법 설계 및 분석. School of Computing . 2015. 34p. Advisor Prof. Kim, Kwangjo. Text in English.

ABSTRACT

To protect the modern IT (Internet technology) system from the malicious adversary, the system is required to equip all security aspects. The most fundamental security aspects are confidentiality, integrity, and availability. Each security aspects are provided by different dedicated algorithms, but the implementation of each dedicated algorithms consumes the system resources. If one security algorithm can provide more than one security aspect efficiently, then the consumption of the system resources can be reduced. An authenticated encryption scheme is a kind of symmetric key cryptosystem to provide confidentiality, integrity, and authenticity at the same time.

Some early authenticated encryption schemes like AES-GCM are standardized and recommended by many standardization organizations like NIST (national institute of standards and technology). However, the research to develop more secure, efficient, and robust authenticated encryption scheme has been sparked recently.

This thesis presents an authenticated encryption scheme based on the duplex construction of the sponge function. The sponge function is a cryptographic primitive, which can be used to build hash functions and authenticated encryption schemes. The proposed authenticated encryption scheme is believed to be competitive compared to other sponge-based authenticated encryption schemes in terms of execution speed, implementation cost, and supportability of the most required authenticated encryption features.

The scheme supports flexible key and tag sizes while increasing the execution speed by maximizing the bitrate of the sponge function and selecting optimal nonlinear and linear layers in the permutation block. The implementation cost is decreased by minimizing the width and the capacity of the sponge function.

The most fundamental security notions like IND-CPA (indistinguishability under chosen-plaintext attack) and INT-CTXT (integrity of ciphertext against forgery) of the authenticated encryption scheme are verified in the security analysis. The proposed scheme is implemented using C++ programming language, and the performance is compared to AES-GCM. Furthermore, the scheme's supportability of the most required authenticated encryption features like nonce-MR (nonce-misuse resistance) is analyzed.

Keywords: Authenticated Encryption, Sponge Function, Symmetric Key Cryptography, Hash Function, CAESAR

Contents

List of Tables

List of Figures

Chapter 1. Introduction

1.1 Overview of Authenticated Encryption

Cryptographic primitives like AES (Advanced Encryption Standard) and SHA (Secure Hash Algorithm) were developed to efficiently provide security requirements like confidentiality and integrity, but each of them ensures only one of the security requirements. However, an authenticated encryption scheme provides confidentiality, integrity, and authenticity at the same time. Authenticated encryption and decryption is stated below:

$$E_k(AD, M, N) \rightarrow (C, T),\ D_k(AD, C, N, T) \rightarrow (M, T) \text{ or } \perp$$

In the equations, $E_k()$ is encryption, $D_k()$ is decryption, k is secret key, AD is associated data (e.g. header), M is message or plaintext, N is nonce, C is ciphertext, T is tag or MAC (message authentication code), and $\perp$ is null.

Early authenticated encryption schemes like AES-CCM [29], AES-GCM [11], OCB [23] use AES as their basic building block and are standardized in many organizations like ISO, IEEE, and NIST. Also, communication protocols like TLS [10] and Zigbee [15] include the above-mentioned authenticated encryption schemes.

Authenticated encryption is either a secure combination of symmetric key cryptosystem and keyed cryptographic hash function or a dedicated security scheme to provide confidentiality, integrity, and authenticity at the same time. Recent researches to develop next-generation authenticated encryption schemes have been focusing on the dedicated authenticated encryption schemes. D. J. Bernstein and other cryptography researchers have launched the CAESAR (Competition for Authenticated Encryption: Security, Applicability, and Robustness) competition [5], which is funded by NIST, to select the next generation authenticated encryption scheme. Many authenticated encryption schemes based on cryptographic primitives like AES, hash function, stream cipher, sponge function, *etc.* are submitted to the first round of the CAESAR competition.

Figure 1.1 illustrates example schematics, and each modes submitted to the CAESAR competitions are illustrated in (a), (b), (c), and (d).

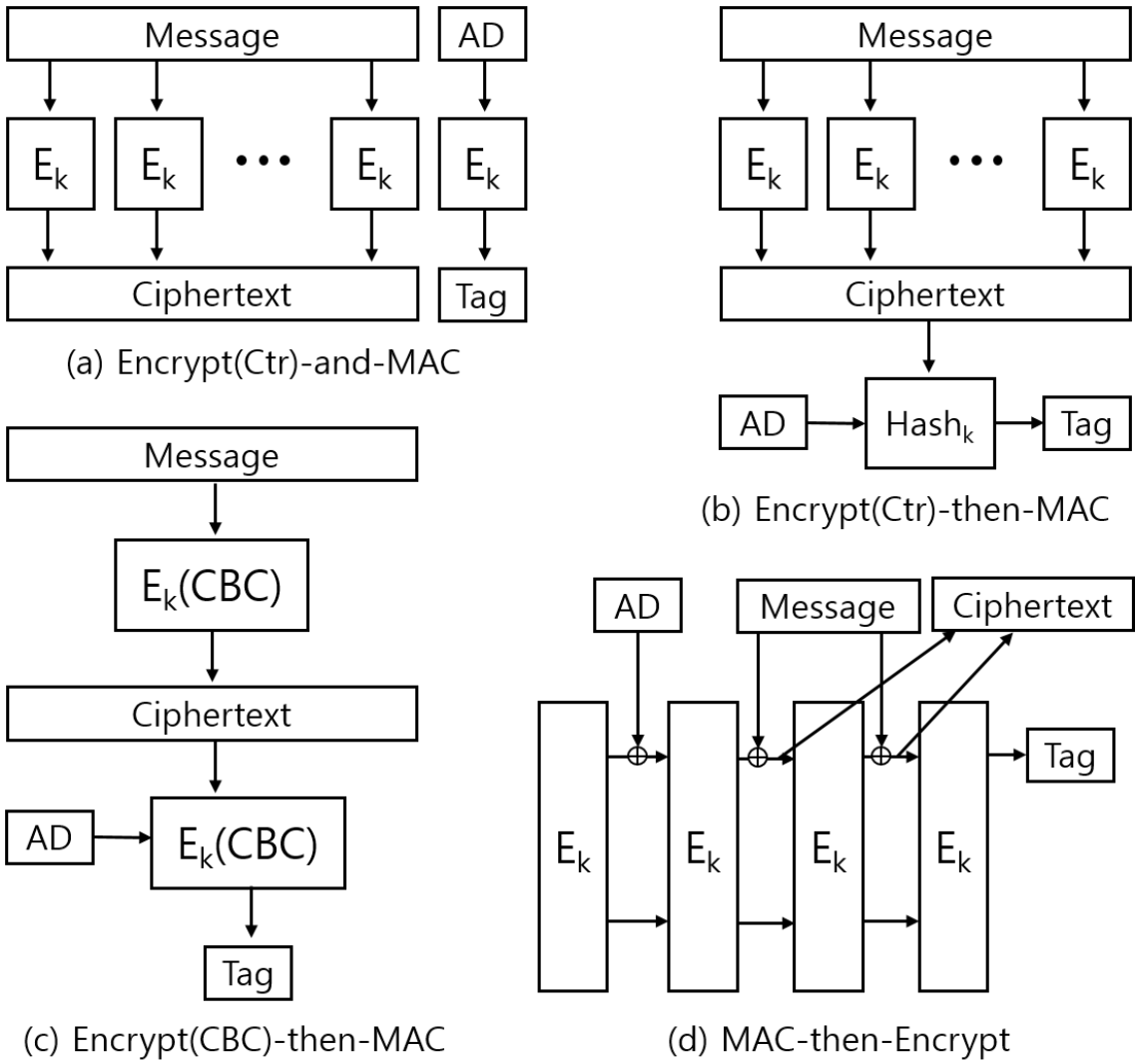

Figure 1.1: Example schematics of authenticated encryption schemes

To evaluate an authenticated encryption scheme, the performance, security, and feature analysis of the scheme are required. In the performance analysis, the execution speed and the implementation cost of the authenticated encryption scheme are compared with those of the existing authenticated encryption schemes like AES-GCM. In the security analysis, the provable security of IND-CPA and INT-CTXT is required, and the resistance of the scheme against various cryptanalysis techniques must be analyzed. In the feature analysis, the supportability of the most required authenticated encryption features is compared with that of other authenticated encryption schemes.

1.2 Motivation

In general, implementing all security requirements to protect the system against the malicious adversary is important. However, the implementation of each security requirements will consume the resources of the system. For example, the installation of IDS (intrusion detection system) will consume a lot of the network resource. Authenticated encryption can reduce the resource consumption by simultaneously providing confidentiality, integrity, and authenticity. Early authenticated encryption schemes like AES-GCM are efficient enough to protect the desktop computers, but the resource requirements of those schemes are too burdensome for the recourse-constrained computing environments like IoT (Internet on things). Therefore, the development of a lightweight authenticated encryption scheme is essential to protect the system efficiently.

To provide a secure authenticated encryption to the system, IND-CPA and INT-CTXT security notions are necessary. IND-CPA security is essential for privacy (confidentiality), and INT-CTXT security is essential for integrity and authenticity. However, some authenticated encryption schemes submitted to the first-round of the CAESAR competition do not provide the provable security of IND-CPA and INT-CTXT.

Through the analysis of the early authenticated encryption schemes and the first-round of the CAESAR competition, many useful features of the authenticated encryption are discussed. Among the features, many researchers agree on the most required authenticated encryption features like inverse-freeness, onlineness, and intermediate tag. Some of the features like inverse-freeness and onlineness contribute to increase the execution speed and to reduce the implementation cost of the authenticated encryption scheme, while other features contribute to increase the security of the scheme. Although the supportability of the most required authenticated encryption features is crucial to build the next-generation authenticated encryption scheme, the existing authenticated encryption schemes and the submissions to the CAESAR competition do not support all of the most required features.

This thesis proposes a novel authenticated encryption scheme based on the duplex construction of the sponge function. The proposed scheme has higher performance than the existing authenticated encryption standards, provides the provable security, and supports all of the most required features.

1.3 Organization

The rest of this thesis is organized as follows: Chapter 2 describes related work and background about the history of the authenticated encryption, the sponge function, and various permutations, which can be used to build a secure cryptographic primitive. The definitions and notations used in this thesis are described in Chapter 3. The design choice, the construction, and the implementation of our scheme are explained in Chapter 4. The performance, security, and feature analysis of the scheme are presented in Chapter 5. Finally, the conclusion and future work are discussed in Chapter 6.

Chapter 2. Related Work and Background

This chapter introduces the key factors that are required to construct the proposed authenticated encryption scheme. Firstly, more detail on the authenticated encryption including the history, the existing authenticated encryption schemes, and the analysis of the submissions to the CAESAR competition is explained. Secondly, we describe the duplex construction of the sponge function, the authenticated encryption schemes based on the sponge function, and the security bound of the sponge function. Furthermore, various permutations, which are used to construct the round function of the sponge function, are discussed.

2.1 Authenticated Encryption

2.1.1 Development of Authenticated Encryption

The authenticated encryption is a combination of two cryptographic primitives; symmetric key cryptography and authentication. In past, implementers simply glued provably secure encryption and authentication algorithms together, but many of the resulting modes were insecure and slow [3]. We define such approach as the **naïve composition** method. The composition methodology of the authenticated encryption can affect the security and the performance. To provide an efficient and secure composition, the researchers proposed three compositions; **general composition paradigm**, **encrypt with redundancy**, and **encode-then-encipher** [3]. General composition paradigm became the winner among three compositions, because encrypt with redundancy and encode-then-encipher were insecure and inefficient compared to general composition paradigm [3][4][2]. However, general composition paradigm has one major disadvantage that two separate keys are required for encryption and authentication.

General composition paradigm is developed into **one-pass** and **two-pass modes**. One-pass mode, which is called **encrypt-and-authenticate (E&M)**, executes encryption and authentication at the same time. Although one-pass mode is not direct descendant of general composition paradigm, one-pass mode is generally faster than two-pass mode. OCB [23] and AES-CCM [29] are examples of one-pass mode. Two-pass mode executes one algorithm first and executes another; **encrypt-then-**

authenticate (EtM) or **authenticate-then-encrypt (MtE)**. Two-mode is a direct descendant of general composition paradigm using one secret key for both encryption and authentication. AES-GCM [11] is an example of two-pass mode authenticated encryption.

One-pass mode and two-pass mode use existing cryptographic primitive like AES, but recent researches are focusing on developing dedicated authenticated encryption schemes. NIST is funding a public competition called **CAESAR** [5] to select the next-generation authenticated encryption scheme. The authenticated encryption schemes submitted to the CAESAR competition are dedicated authenticated encryption schemes or are based on various cryptographic primitives. Although the CAESAR competition is not official standardization competition hosted by NIST, the competition has drawn global attention. The development of the authenticated encryption is illustrated in Figure 2.1. Encryption with redundancy and encode-then-encipher are crossed out, because they lost the competition with general composition paradigm.

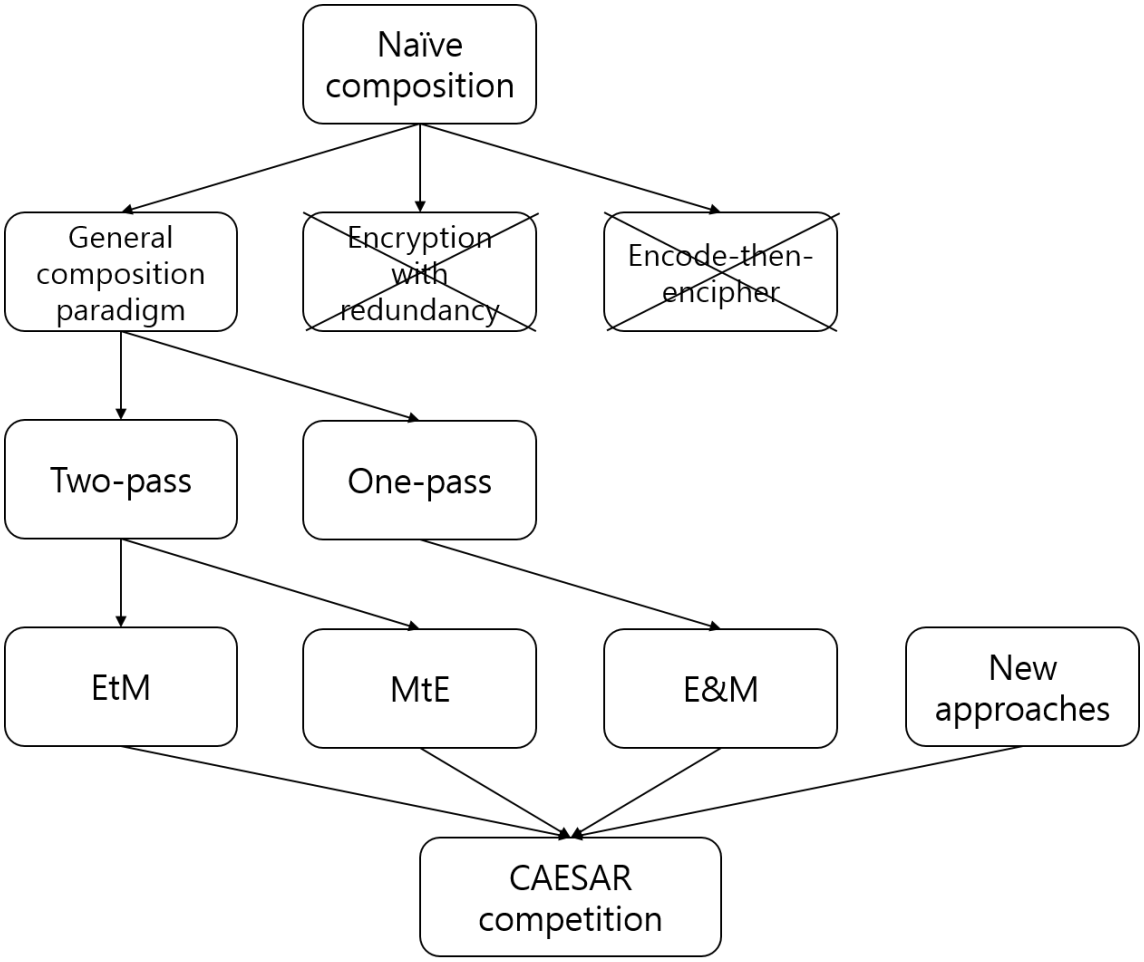

Figure 2.1: The development of authenticated encryption

2.1.2 Existing Authenticated Encryption Schemes

AES-CCM

AES-CCM (counter with CBC-MAC) [29] is a standard mode of operation which provides both encryption and authentication of given data. The scheme is an example of E&M mode, which can execute encryption and authentication at the same time. Figure 2.2 illustrates the structure of AES-CCM. In encryption part, the generated counter is encrypted, and the result is exclusive-ored with plaintext to generate ciphertext. In authentication part, nonce, associated data, and plaintext are concatenated and encrypted in CBC (cipher block chaining) mode to generate tag. In other words, encryption is processed via counter mode, and authentication is processed via CBC-MAC mode. The scheme has limited supportability of the most required authenticated encryption features. Features like onlineness, intermediate tag, incremental AD/AE, nonce-MR, and decryption-MR are not supported in AES-CCM. We list and explain the most required features in Section 2.1.3. Furthermore, encryption is parallelizable, but authentication must be executed serially. AES-CCM does not have good performance, and some researchers criticized the security of AES-CCM.

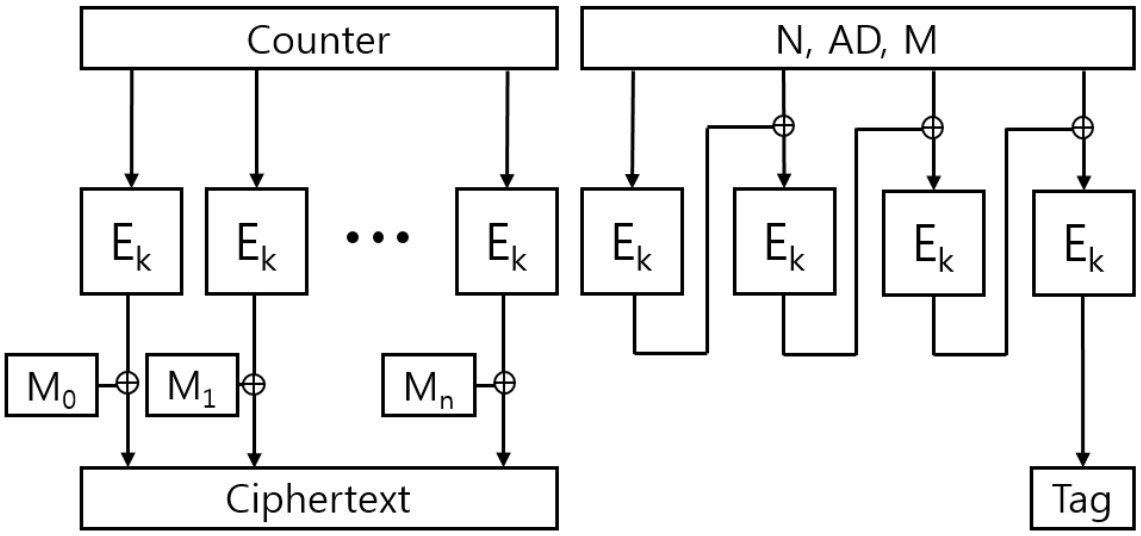

Figure 2.2: The structure of AES-CCM

AES-GCM

AES-GCM (Galois / Counter Mode) [11] is also a standard mode of operation which provides both encryption and authentication of given data. The scheme is an example of EtM mode, which should execute encryption and authentication consequently. The encryption part is similar to that of

AES-CCM. However, the authentication part is processed via universal hashing under binary galois field using nonce, associated data, and ciphertext. The scheme and AES-CCM have similar supportability of the most required features. However, AES-GCM has better performance and security than AES-CCM. Figure 2.3 illustrates the structure of AES-GCM.

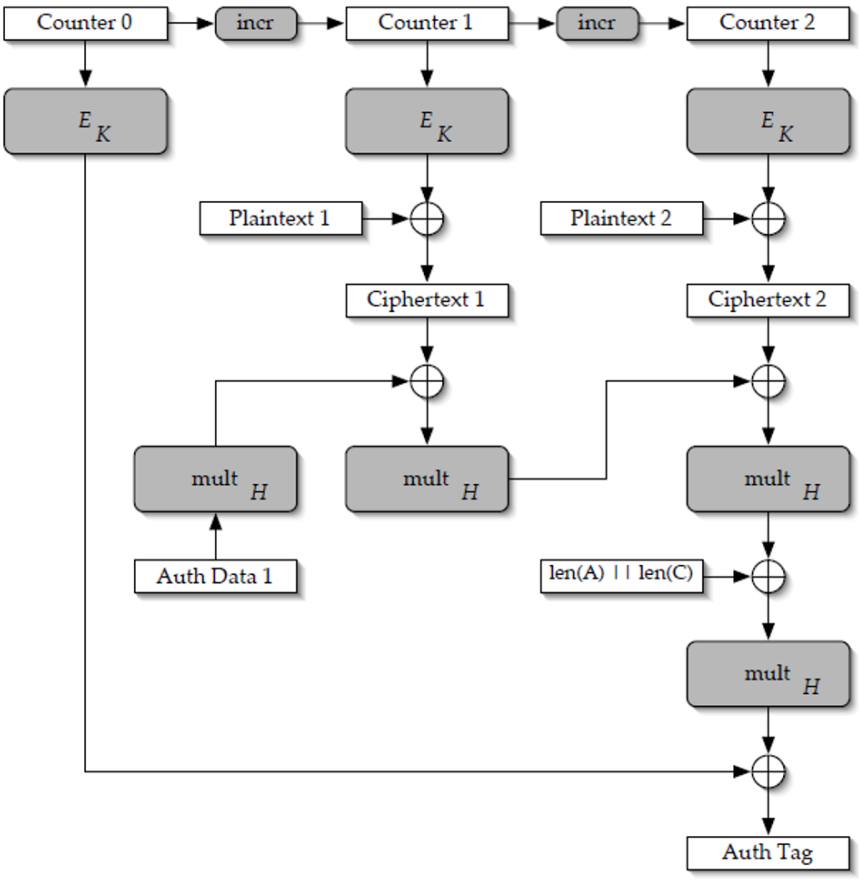

Figure 2.3: The structure of AES-GCM [11]

OCB

OCB (offset codebook) [23] is a mode of operation which provides encryption and authentication in parallel. Like AES-CCM, the scheme is an example of E&M mode. In both encryption and authentication parts, an encryption algorithm like AES is used with input and output whitening. OCB calculates input and output whitening variable using a Gray code and encryption result of nonce, and exclusive-ors the input and the output of encryption and authentication with the variable. The encryption part and the authentication part are fully parallelizable to boost the performance of OCB. The scheme is online and provides the provable security, but OCB does not support many features like intermediate tag and

incremental AD/AE. Due to its parallelizability, OCB has better performance than AES-GCM and AES-CCM, and the scheme has higher security level than AES-CCM in terms of indistinguishability and non-malleability under CCA (chosen-ciphertext attack). The structure of OCB is illustrated in Figure 2.4.

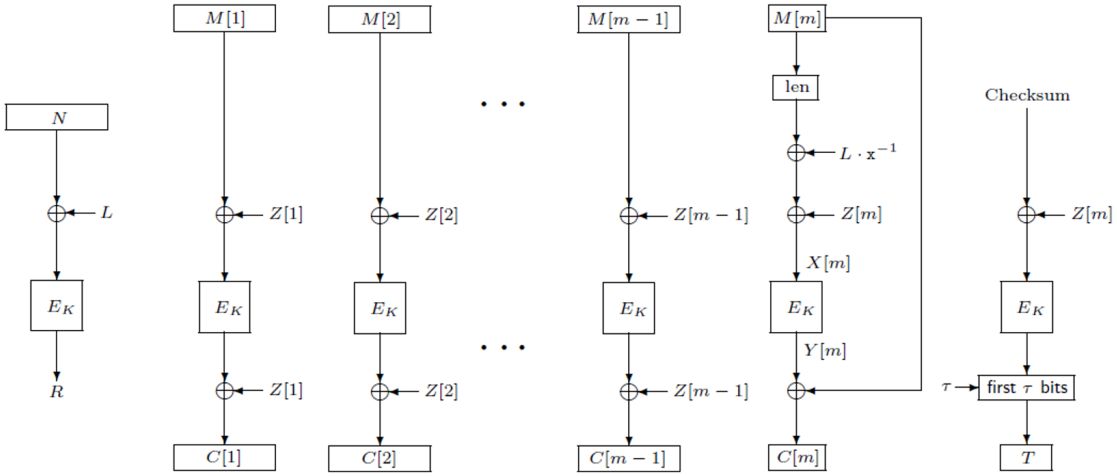

Figure 2.4: The structure of OCB [23]

2.1.3 Analysis of the CAESAR submissions

Abed *et al.*[1] classified the authenticated encryption schemes submitted to the CAESAR competition. In [1], many authenticated encryption features are introduced, and the overview of the features and provable security for all submissions to the CAESAR competitions is presented. In Figure 2.5, we illustrate a part of the overview from [1].

Bellare and Namprempre [3] proved that IND-CPA and INT-CTXT are necessary to consider an authenticated encryption scheme to be secure. Andreeva *et al.* proposed a new security notion, INT-RUP (integrity of ciphertext under releasing unverified plaintext setting), for authenticated encryption schemes based on practically unavoidable scenarios in which the schemes may release unverified plaintext due to the insufficient memory to store entire plaintext or that the system requires early processing of the plaintext. INT-RUP is renamed to be a feature called decryption-MR in [1].

According to [1], the most required authenticated encryption features are parallelizability, onlineness, inverse-freeness, incremental AD/AE (incremental associated data and authenticated encryption processing), intermediate tag, fixed AD reuse, provable security, nonce-MR (nonce-misuse resistance),

and decryption-MR. We consider that 'no ciphertext expansion' feature is also important. Parallelizability means that the execution of the underlying cryptographic primitives is parallelizable. Onlineness means that the number of the underlying cryptographic primitive calls equals to the number of nonce, associated data, and plaintext blocks. Inverse-freeness means that the scheme does not require inverse function of the underlying primitive to decrypt the ciphertext. Incremental AD/AE means that the scheme is able to process a small fraction of associated data or plaintext without re-initialization after encrypting lots of data. Fixed AD reuse means that the scheme caches and reuses the processed result of associated data to boost the performance. Nonce-MR means that the scheme is secure when the implementation of the scheme does not guarantee the uniqueness of nonce. No ciphertext expansion means that the size of plaintext equals to the size of ciphertext. In this thesis, we aim to provide all features mentioned above.

Construction	**Candidate**	**Design**	**Primitive**	**Features**						**Security**		
				Parallelizable Enc/Dec	*Online*	*Inverse-Free*	*Incremental AD/AE*	*Fixed AD reuse*	*Intermediate Tags*	*Security proof*	*Nonce-MR*	*Decryption-MR*
Feistel-state-based	AES-AEGIS [78]	–	AES-round	•/–	•	•	–/–	–	–	–	–	–
	MORUS [77]	–	MORUS	–/–	•	•	–/–	–	–	–	–	–
	Tiaoxin [60]	–	AES-round	•/•	•	•	–/–	–	–	–	–	–
Stream-cipher-based	ACORN [75]	–	ACORN	•/•	•	•	–/–	–	–	–	–	–
	Calico [71]	–	ChaCha, SipHash	–/–	•	•	–/–	–	–	–	–	–
	Enchilada [37]	–	ChaCha, Rijndael	•/•	•	•	•/–	•	–	•	–	–
	HS1-SIV [48]	SIV	ChaCha, Poly1305	–/–	–	•	–/–	–	–	•	•	–
	Raviyoyla [73]	–	MAGv2	–/–	•	•	–/–	–	–	–	–	–
	Sablier [81]	–	Sablier	•/•	•	•	•/–	•	–	–	–	–
	TriviA-ck [22]	–	Trivia-SC	•/•	–	•	–/–	–	•	•	–	–
	Wheesht [54]	–	Wheesht	–/–	•	•	–/–	–	–	–	–	–

Figure 2.5: The overview of CAESAR submissions [1]

2.2 Overview of the Sponge Function

In this Section, we describe the well-known cryptographic primitive called the sponge function. The primitive became famous after Keccak [21], which is based on the sponge function, won the SHA-3 competition. The sponge function is an efficient, secure, and versatile cryptographic primitive used in hash function and authenticated encryption. We will present a brief explanation of the sponge function in Section 2.2.1 and the duplex construction of the sponge function in Section 2.2.2.

2.2.1 Sponge Function

The sponge function is a function that maps an arbitrary length input to an arbitrary length output. The operation of the sponge function applies an inner permutation to a fixed length state iteratively. The size of a state is called the width, and the width is a concatenation of the bitrate and the capacity. The input and the output of the sponge function is operated on the bitrate, and the capacity should be never directly interact with the input and the output. The sponge function has two phases; absorbing phase and squeezing phase. In the absorbing phase, the arbitrary length input is exclusive-ored with the bitrate. In the squeezing phase, the bitrate is returned as the output. The basic structure of the sponge function is illustrated in Figure 2.6. M is the message, r is the bitrate, c is the capacity, f is a cryptographic primitive, and Z is the result (hash value or keystream). The sponge function can be used to build hash functions and stream ciphers. In [7], the security bound of the sponge function was proved to be $\min\{2^{c/2}, 2^k\}$. k is the size of the key.

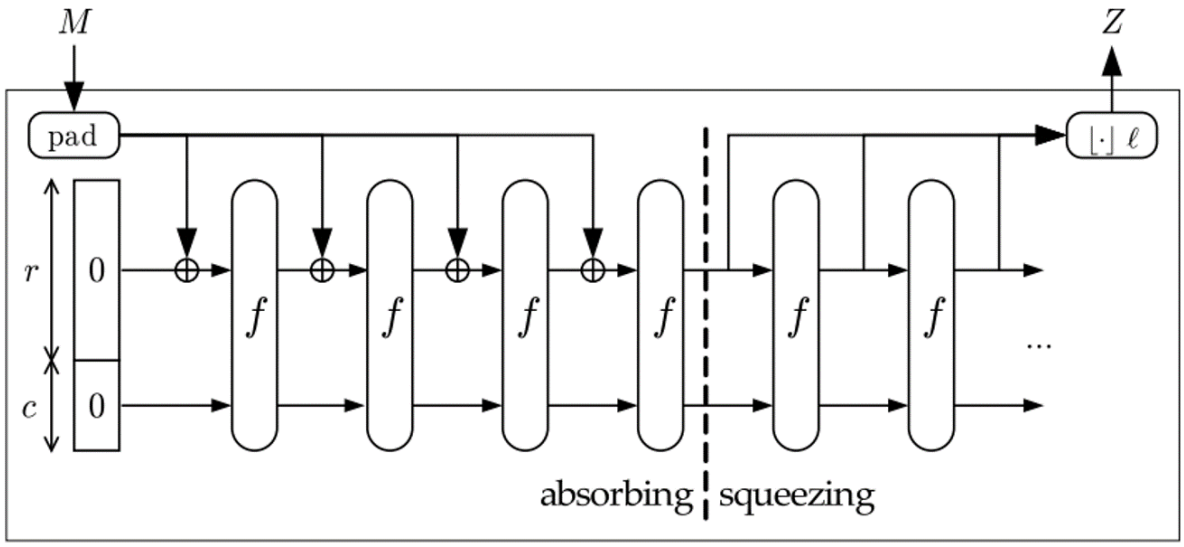

Figure 2.6: The structure of sponge function [7]

2.2.2 Duplex Construction

The duplex construction of the sponge function is a special construction of the sponge function dedicated to implement the authenticated encryption. Similar to the sponge function, the duplex construction takes an arbitrary length input to generate an arbitrary length output. However, the duplex construction does not have the absorbing phase and the squeezing phase. The construction has only one phase; duplexing phase. In the duplexing phase, the input and the output alternates between the underlying permutation calls. The duplex construction of the sponge function naturally supports onlineness, inverse-freeness, and incremental AD/AE. Most of the authenticated encryption scheme based on the duplex construction uses two phases; absorbing phase and duplexing phase. The associated data blocks are processed in the absorbing phase, and the encryption is executed in the duplexing phase. The basic structure of the duplex construction is illustrated in Figure 2.7. σ_i is i^{th} message block.

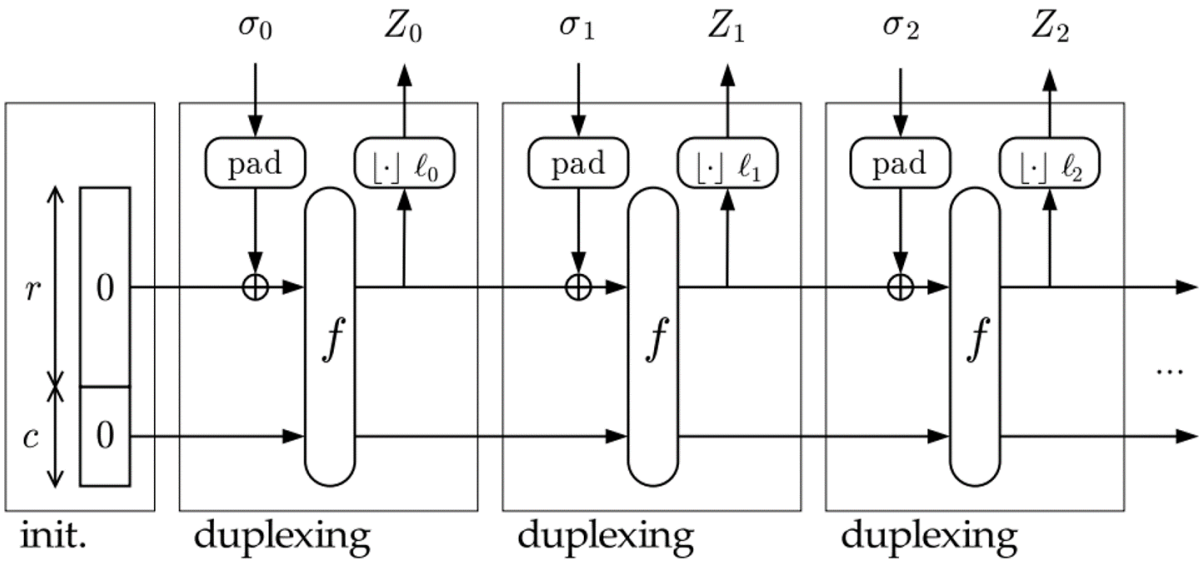

Figure 2.7: The structure of duplex construction [8]

The security of the duplex construction is inherited from the security of the sponge function [8], so the security bound of the duplex construction is known to be $\min\{2^{c/2}, 2^k\}$. However, P. Jovanovic *et al.* [12] proved that the security bound of the authenticated encryption schemes based on the duplex construction should be increased to $\min\{2^c, 2^{b/2}, 2^k\}$. b is the size of the width. Thus, the size of the capacity can be reduced to increase the size of the bitrate, and larger bitrate can take more input bits and generate more output bits at one time. The performance of the scheme will be increased.

The detailed version of the security bound from [12] is stated below:

$$\mathbf{Adv}_{\Pi}^{\mathbf{priv}}(q_p, q_\epsilon, \lambda_\epsilon) \leq \frac{3(q_p+\sigma_\epsilon)^2}{2^{b+1}} + (\frac{8eq_p\sigma_\epsilon}{2^b})^{1/2} + \frac{rq_p}{2^c} + \frac{q_p+\sigma_\epsilon}{2^\kappa} \text{ and}$$

$$\mathbf{Adv}_{\Pi}^{\mathbf{auth}}(q_p, q_\epsilon, \lambda_\epsilon, q_D, \lambda_D) \leq \frac{(q_p+\sigma_\epsilon+\sigma_D)^2}{2^b} + (\frac{8eq_p\sigma_\epsilon}{2^b})^{1/2} + \frac{rq_p}{2^c} + \frac{q_p+\sigma_\epsilon+\sigma_D}{2^\kappa} + \frac{(q_p+\sigma_\epsilon+\sigma_D)\sigma_D}{2^c} + \frac{q_D}{2^\tau}.$$

Adv is the advantage of the adversary. **priv** is the privacy, which is IND-CPA (indistinguishability under chosen-plaintext attack). **auth** is the authenticity, which is INT-CTXT (integrity of ciphertext). Π is an random oracle. $\mathbf{Adv}_{\Pi}^{\mathbf{priv}}$ is the advantage of the adversary to distinguish the output of Pi and the output of the encryption. $\mathbf{Adv}_{\Pi}^{\mathbf{auth}}$ is the advantage of the adversary to successfully forge the authentication tag. q_p is the number of queries by the adversary to the permutation block. q_ϵ is the number of encryption queries. λ_ϵ is the number of plaintext blocks used in the queries. σ_ϵ is the number of primitive calls in encryption. q_D is the number of decryption queries. λ_D is the number of ciphertext blocks used in the queries. σ_D is the number of primitive calls in decryption. κ is the size of the key, and τ is the length of the tag.

2.3 Permutations

The security of the sponge function is proved rigorously under an assumption that the underlying round function, also called as permutation block, is secure. The round functions of all modern cryptosystems consist of nonlinear and linear layers. There are one nonlinear layer and three linear layers in our round function. Todo *et al.* [27] classified different linear layers into three categories, which are bP (bit Permutation), BP (Byte Permutation), and MP (Matrix Permutation), and claim that bP and MP are essential to construct the perfect diffusion [13]. Three linear layers in the proposed scheme cover all three categories proposed in [27].

Multipermutation

The serious research on the diffusion (linear) layers used in modern cryptography was started from the introduction of the term, multipermutation by Schnorr and Vaudenay [24][25]. Mileva [17] connected the generalized concept of multipermutation [28] and other permutations. We briefly recall the definition as below:

Definition 1 (Multipermutation) [28]. *A function* $f : Z^{\rho} \rightarrow Z^{\gamma}$ *is an* (ρ, γ)*-multipermutation if any two different* $(\rho + \gamma)$ *-tuples of the form* $(x_1, x_2, x_3, \ldots, x_{\rho}, f(x_1, x_2, x_3, \ldots, x_{\rho}))$ *cannot collide in any* ρ *positions.*

Multipermutation achieves the perfect diffusion, because "changing i input values on $f(x_1, x_2, x_3, \ldots, x_{\rho})$ will change at least $(\gamma - i + 1)$ output values" [17]. Thus, changing 1 input value in $(2, 2)$-multipermutation will change at least 2 output values.

Latin square

According to [17], "a quasigroup is a groupoid $(Q, *)$ that satisfies the property for each one of the equations $a * x = b$ and $y * a = b$ to have a unique solution x, respectively y". Latin square is equivalent to a $(2, 1)$-multipermutation, and a pair of orthogonal latin square is equivalent to $(2, 2)$-multipermutation. Ristenpart and Rogaway [22] used a pair of orthogonal latin square to securely mix two strings efficiently, and provided a simplified version of the pair of orthogonal latin square to increase the performance while the security difference is negligible. The schematic of their mixing function is illustrated in Figure 2.8. The operation **dbl**, used in the original pair of orthogonal latin square, is a multiplication by 2 in the finite field F_2, and the operation **rol**, used in the simplified version, is an 1-bit circular left rotation. We use the simplified version in the proposed authenticated encryption scheme.

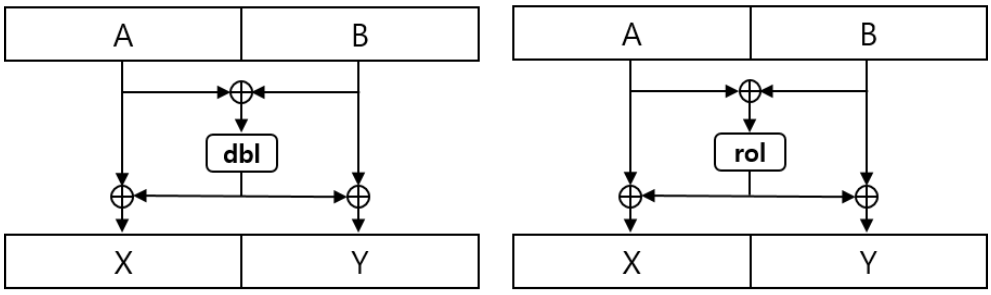

Figure 2.8: The schematic of mixing function [22]

MDS

MDS (Maximum Distance Separable) code is a linear code that meets the singleton bound. Under a finite field, a $[\rho + \gamma, \rho, \gamma + 1]$ MDS code is equivalent to a (ρ, γ)-multipermutation [17]. Many modern cryptosystems use MDS matrix in the linear layer. AES (Advanced Encryption Standard) uses 4x4 MDS circulant matrix [17], also called as *MixColumn()*, to diffuse the state. $(4, 4)$-multipermutation can be

represented by 4x4 MDS matrices, and the word size in the matrices can be 8, 16, and 32-bit. There have been many researches to build efficient MDS matrices in software or hardware. Especially, Junod and Vaudenay [13] gave software-efficient constructions of 4x4 and 8x8 MDS matrices. We use a 4x4 MDS matrix in the proposed scheme.

Linear layer in PRESENT

PRESENT [9] is an well-known lightweight block cipher optimized for the hardware implementation. The block size of PRESENT is 64-bit and the supported key sizes are 80-bit and 128-bit. The round function of PRESENT consists of one nonlinear layer and one linear layer. The linear layer permutes 64-bit state using a table. The proposed authenticated encryption scheme includes a linear layer that modifies the linear layer of PRESENT using 64-byte state. Because the linear table of PRESENT does not change four bit positions from input state to output state, we modify the table to ensure that all byte positions from input state to output state are mixed.

Chapter 3. Preliminaries

In this chapter, we state and explain the notations, basic subroutines, and various definitions used in this thesis.

3.1 Notations

Following notations are used in this thesis, and any other notations or variables not stated in this section are explained in the corresponding sections. Table 3.1 describes the notations.

Table 3.1: Notations and Variables

Variables	Description
pt	whole plaintext, or message, in an instance
pt_i	i^{th} block of plaintext
ct	whole ciphertext in an instance
ct_i	i^{th} block of ciphertext
ad	whole associated data in an instance
ad_i	i^{th} block of associated data
k	secret key
tag	equivalent to message authentication code (MAC)
it	intermediate tag
$E_k(pt, ad, Nonce)$	authenticated encryption of pt using ad and $Nonce$
$D_k(pt, ad, Nonce, tag)$	authenticated decryption of ct using ad, $Nonce$, and tag
r	bitrate of sponge function
c	capacity of sponge function
b	width of sponge function
S	b-bit state of sponge function
$\|\|$	concatenation operator
$\oplus$	bitwise exclusive-or operator
$\|string\|$	bit-length of a $string$

3.2 Subroutines

The subroutines stated in Table 3.2 are basic functions used in the pseudocode of the proposed authenticated encryption schemes. The simplified version of the pseudocode is presented in Appendix A, and the full implementation is presented in Appendix B.

Table 3.2: Subroutines

Subroutine	Description
$truncate()$	truncate and extract given length of bits starting from MSB (most significant bit) from S
$divideStr()$	divide the given variable-length string y the size of the block, bitrate r. The first pt/ct block is $\|r\| - \|it\|$ long. The rest of the pt/ct will be divided by $\|r\|$. Any leftover bits from the string will be padded using subroutine $pad()$
$pad()$	pad $1\overbrace{000...}^{*}1$ bits to make plaintext or associated data divisible by $\|r\|$. This is called a sponge-compliant multi-rate padding [8]
$unpad()$	unpad bits, which bit-length is equal to the number of bits padded to plaintext in encryption
$init()$	initialize the state S with a random number, exclusive-or k and $Nonce$ to S, and apply $latinSquare()$ to S
$latinSquare()$	return $(A \oplus rol_1(A \oplus B))\|\|(B \oplus rol_1(A \oplus B))$
$rol_n()$	apply left circular n-bit rotation to the string

3.3 Definitions

Bellare and Namprempre [3] defined the most fundamental security notions of the authenticated encryption, and Abed *et al.* derived that IND-CPA and INT-CTXT are necessary to consider an authenticated encryption scheme to be secure. We briefly restate the definitions as below:

Definition 2 (IND-CPA security) [1]. *Let* $\Pi = (K, E, D)$ *be an authenticated encryption scheme. Then, the IND-CPA-advantage of a computationally bounded adversary A for* Π *is defined as*

$$Adv_{\Pi}^{IND-CPA}(A) \leq \left(Pr\left[\kappa \xleftarrow{\$} K, A^{E(\cdot,\cdot)} \Rightarrow 1)\right] - Pr\left[A^{\$(\cdot,\cdot)} \Rightarrow 1\right]\right).$$

This is the maximum advantage for all IND-CPA-adversaries A on Π *that run in time at most t, and make at most q queries of total length l to the available oracles.*

Definition 3 (INT-CTXT security) [1]. *Let* $\Pi = (K, E, D)$ *be an authenticated encryption scheme. Then, the INT-CTXT-advantage of a computationally bounded adversary A for* Π *is defined as*

$$Adv_{\Pi}^{INT-CTXT}(A) \leq Pr\left[\kappa \xleftarrow{\$} K, A^{E(\cdot,\cdot),D(\cdot,\cdot)} \Rightarrow forges\right].$$

This is the maximum advantage for all INT-CTXT-adversaries A on Π *that run in time at most t, and make at most q queries of total length l to the available oracles.*

Chapter 4. Our Proposed Scheme

This chapter proposes a new scheme, which combines the related works stated in Chapter 2.

4.1 Design Choices

This section explains our design choices on the overall structure and the round function of the proposed scheme.

4.1.1 Sponge Function

The sponge function introduced by Bertoni *et al.* [6] is a novel construction method to build hash functions and authenticated encryption schemes efficiently. The superiority of the sponge function is proven when Keccak awarded the SHA-3 competition [21] organized by NIST. Many cryptography researchers have been developing new hash functions and authenticated encryption schemes like ICEPOLE [18] based on the sponge function. To design a novel lightweight authenticated encryption scheme, we find the following characteristics of the sponge function attractive: The security of the sponge function is proven rigorously. Algorithms based on the sponge function exhibit high software/hardware performance. Authenticated encryption schemes based on the sponge function must be inverse-free. The most required features of authenticated encryption schemes can be easily implemented using the sponge function.

The provable security is a necessity in modern cryptography, and the provable security of the sponge function allows to build secure authenticated encryption schemes efficiently. The high performance and inverse-freeness of the sponge function makes the resulting authenticated encryption scheme to be lightweight. Furthermore, we discovered that important features of authenticated encryption schemes like incremental AD/AE, intermediate tag, and nonce-MR can be easily and efficiently implemented using the sponge function.

4.1.2 Round Function

The design of provably secure and efficient round function is essential to build a secure authenticated encryption scheme based on the sponge function. The round function of all modern cryptography

consists of the nonlinear layer and the linear layer. S-box is commonly used to construct the nonlinear layer in modern symmetric key cryptosystem. Thus, the security and the efficiency of S-box have been extensively researched. The efficiency of S-box is widely studied, because the implementation of nonlinear layer is slower and requires more cost than the implementation of linear layer. Some S-boxes are designed to minimize the hardware implementation cost while other S-boxes are designed to maximize the software performance. At this stage, our nonlinear layer is implemented using 8-bit S-box of AES, which provides maximum nonlinearity.

The linear layer diffuses output bits of S-boxes, so the design of the linear layer is important to ensure the overall security of a cryptosystem. Although the linear layer is relatively not studied intensively compared to the nonlinear layer, Vaudenay [28] proposed the notion of "the perfect diffusion". Todo *et al.* [27] classified different permutations into bP, BP, and MP, and proposes that bP and MP is necessary to achieve the perfect diffusion. Therefore, we choose three linear layers to cover bP, BP, and MP to build the perfect diffusion layer in an efficient manner.

4.1.3 Other important design choices

The researchers of authenticated encryption are interested in many features. The features are parallelizability, onlineness, inverse-freeness, incremental AD/AE, intermediate tag, nonce-MR, and decryption-MR [1]. Among those features, we focus on onlineness, inverse-freeness, incremental AD/AE, intermediate tag, nonce-MR, and decryption-MR. Onlineness, inverse-freeness, and incremental AD/AE contribute to less implementation cost and faster execution speed of authenticated encryption schemes, and the features are easily implemented using the sponge function. Intermediate tag allows the system to verify the encrypted message before decrypting the ciphertext, and can be used to increase the performance and protect the authenticated encryption from side-channel attacks. The use of intermediate tag automatically supports decryption-MR feature. Nonce-MR prevents the advantage given to adversaries when a careless implementation of a cryptosystem uses a fixed and repeated nonce to encrypt a message. We consider an efficient implementation of intermediate tag and nonce-MR in our authenticated encryption scheme.

4.2 Overview of the Scheme

Our proposed authenticated encryption scheme is based on the duplex construction of the sponge function. The proposed authenticated encryption scheme takes an advantage from the research of Jovanovic *et al.* [12]. Jovanovic *et al.* proved that security bound of authenticated encryption schemes based on the sponge function is $\min\{2^{b/2}, 2^c, 2^k\}$, which is significantly higher than the $2^{c/2}$ security of the sponge function. To boost the performance of the scheme, we reduce the capacity c to increase the bitrate r. When the bitrate r is increased, the number of bits from associated data and plaintext can be processed more in each execution of the permutation block. In following subsections, we describe the parameters, the overall process of the proposed scheme, and the detailed structure of the round function.

4.2.1 Parameters and Overall Process

Our scheme is designed to support the flexible key size. The key size can be changed to meet the different security requirements. The recommended key sizes of the proposed scheme are 128, 192, and 256-bit. To minimize the bit-size of the state S in the sponge function, we set the minimum size of $b/2$ to be equal to the maximum size of c and k. Thus, the minimum size of the state S will be 512-bit. The recommended size of secret key k will be 128, 192, and 256-bit, the recommended size of capacity c will be 128, 192, and 256-bit, and the recommended size of bitrate r will be 384, 320, and 256-bit, respectively. The maximum length of tag is equal to that of k. The summary of the recommended parameters of the proposed authenticated encryption scheme is presented in Table 4.1.

Table 4.1: Parameters

k	r	c	b	tag	**Security bound**	**Number of rounds**
128	384	128	512	0∼128	128	5
192	320	192	512	0∼192	192	6
256	256	256	512	0∼256	256	8

The overall execution of the proposed scheme is described as follows: (1) Initialize state S and apply round function. (2) Process associated data ad. (3) Extract intermediate tag it from r. (Optional) (4) Process plaintext pt and generate ct. (4) Extract tag and $nextNonce$ (Optional) from S.

The 128-bit version of the proposed scheme is illustrated in Figure 4.1. Note that index i in P_i means the number of round repetition in the permutation block P_n. The differences between our scheme and other authenticated encryption schemes based on the sponge function are as follows: (1) initialization function diffuses k and $Nonce$ using *latinSquare()*. (2) intermediate tag, it is generated before the encryption. (3) $nextNonce$ is generated at the end of the scheme.

At the first execution, the sender must generate and use $Nonce$ for the authenticated encryption and send $Nonce$ along with ct, ad, tag to the receiver. When it is extracted, the leftover bits of r will be exclusive-ored with the plaintext of same size. The value of S is initialized to the 512-bit random number provided by NIST Randomness Beacon [20] at 11:12 AM on April 1st, 2015. The random number is 0x20AAF8DDB4E40B2814B93E0B57BBB336777DAF8B60BACF06A131EB7E0D8C923CD184E3A1C1 031A3B3AFBC89601CAD91CC28C6DA918A7E4E37DB46EED18A9607C. The state value is initialized to a random number to resist the rotational cryptanalysis as shown in Section 5.2.2.

The supported authenticated encryption features including intermediate tag and nonce-MR are explained in below: (1) Onlineness and inverse-freeness features – The authenticated encryption scheme based on the duplex construction of the sponge function is naturally online and inverse-free. (2) Incremental AD/AE feature – The sender may process few blocks of associated data or plaintext using the state of the previous execution. The sender does not need to send any other information to process the incremental AD/AE. (3) Intermediate tag and decryption-MR features – The sender may send it for the receiver to verify the message before decrypting ct. In this case, the system does not release unverified plaintext and achieves decryption-MR. (4) Nonce-MR feature – At subsequent executions, the sender may use $nextNonce$, which is generated at the previous executions, respectively. The sender does not need to send $nextNonce$ to the receiver when using $nextNonce$ in subsequent executions, because the receiver should know the value of $nextNonce$ from the state of the previous executions. (5) No ciphertext expansion feature – pt, ad, and ct are padded and unpadded during the execution. The length of ct transmitted to the receiver is equal to the length of the original pt.

4.2.2 Details of Round Function

The duplex construction of the sponge function used in the authenticated encryption scheme is proven to be secure if the underlying permutation block P_n of the scheme is secure [8]. Our P block

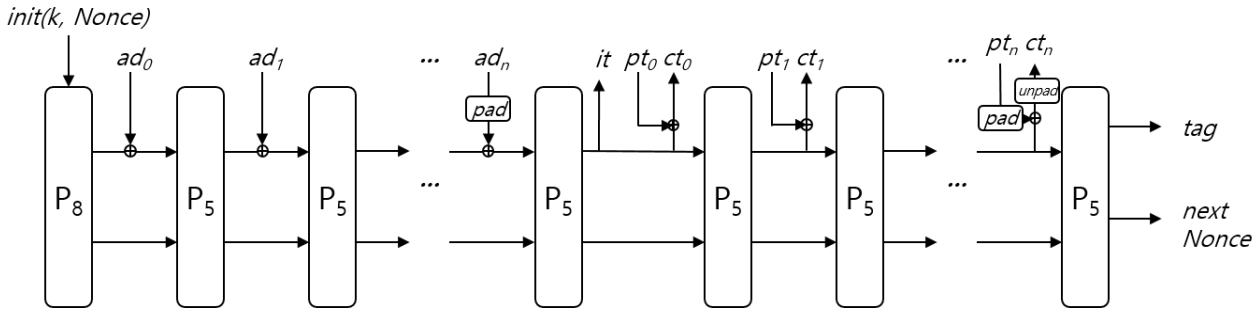

Figure 4.1: The Proposed Scheme

is constructed via the repetition of the round function. The round function of modern cryptography consists of the nonlinear layer and the linear layer. Thus, the choice and the combination of the layers are essential to provide the security to our proposed authenticated encryption scheme. In this subsection, we describe the design of the round function in our proposed scheme in detail.

S-box is used as the nonlinear layer in most modern cryptosystems including the sponge function. S-box substitutes a fixed number of input bits to the fixed number of output bits, and the number of input bits and output bits is called the size of the S-box. The nonlinear layer of our round function is constructed using S-box. We select a maximally nonlinear 8-bit S-box from AES, because the nonlinear layer consumes the most resource in the symmetric key cryptosystem. 8-bit S-box from AES has lower differential/linear characteristic/approximation than any 4-bit S-boxes, so we can use less number of rounds to resist the differential/linear cryptanalysis. Although 8-bit S-box has proportionally higher implementation cost than 4-bit S-box [16], the use of 8-bit S-box increases the execution speed by parallel byte-processing of 8-bit S-box and by allowing less number of rounds. In addition, any 8-bit S-box with the best security characteristics like differential/linear characteristic/ approximation can be adopted to protect the authenticated encryption scheme from linear and differential cryptanalysis.

The linear layer is also called the diffusion layer, which concept was first introduced by Shannon [26]. The linear layer dissipates the statistical structure of a string [17], and is an essential building block of round functions in modern symmetric key cryptosystems. The linear layer plays an important role in symmetric key cryptosystems to diffuse the output of the nonlinear layer, so that the output bits of an S-box will enter different S-boxes in subsequent rounds.

The size of the state S is 512-bit and the size of our S-boxes are 8-bit. Thus, we need 64 S-boxes in our nonlinear layer. We must make sure that output of an S-box in a round will not enter the

same S-box in next round, so the modified linear layer of the PRESENT [9] is used as BP in our linear layer. We first apply left 4-bit circular rotation to the state and apply BP. The original linear layer of PRESENT changes every single bit position of 64-bit state, so we choose to change every single byte position, instead of bit position, of our 512-bit state. We modified original PRESENT linear layer to ensure that the output of an S-box will enter at least two different S-boxes in the next round. The BP part of our linear layer is stated in Table 4.2.

Table 4.2: Modified Linear Layer of PRESENT

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
B(x)	21	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
x	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
B(x)	4	20	36	52	5	42	37	53	6	22	38	54	7	23	39	55
x	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
B(x)	8	24	40	56	9	25	41	57	10	26	63	58	11	27	43	59
x	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
B(x)	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	0

According to Todo et al. [27], using only BP in the linear layer of symmetric key cryptography is not secure. Thus, we add two multipermutations, classified as bP and MP in [27], to our linear layer. Orthogonal latin square is one of the multipermutation [17], and Ristenpart and Rogaway [22] proposed the use of a pair of mutually orthogonal latin squares and proved that the difference in security level of the simplified version with the original orthogonal latin squares is negligible. The subroutine, *latinSquare()*, is the implementation of the simplified version. The simplified version has a performance advantage over the original version. Thus, we choose to apply the simplified version of a pair of mutually orthogonal latin squares as bP in our linear layer.

The last part of our linear layer is MP. One of the most frequently used MP is MDS, and Mileva [17] proved that MDS is a multipermutation. Many existing symmetric key cryptosystems use MDS in their linear layer, and, for example, *MixColumn()* of AES uses a 4x4 MDS circulant matrix [17]. There have been many researches to build efficient MDS matrices in software/hardware. Especially, Junod and Vaudenay [13] gave software-efficient constructions of 4x4 and 8x8 MDS matrices. Although the inverse matrices of the obtained matrices are not efficient in software, the inverse-free structure of our proposed authenticated encryption scheme does not require the inverse matrices to decrypt ct. We use 4x4 MDS matrix efficient in software from FOX64 [14], and the matrix is stated in Figure 4.2. The schematic of

the round function is illustrated in Figure 4.3.

$$\begin{pmatrix} 0x01 & 0x01 & 0x01 & 0x02 \\ 0x01 & 0xFD & 0x02 & 0x01 \\ 0xFD & 0x02 & 0x01 & 0x01 \\ 0x02 & 0x01 & 0xFD & 0x01 \end{pmatrix}$$

Figure 4.2: MDS from FOX64 [14]

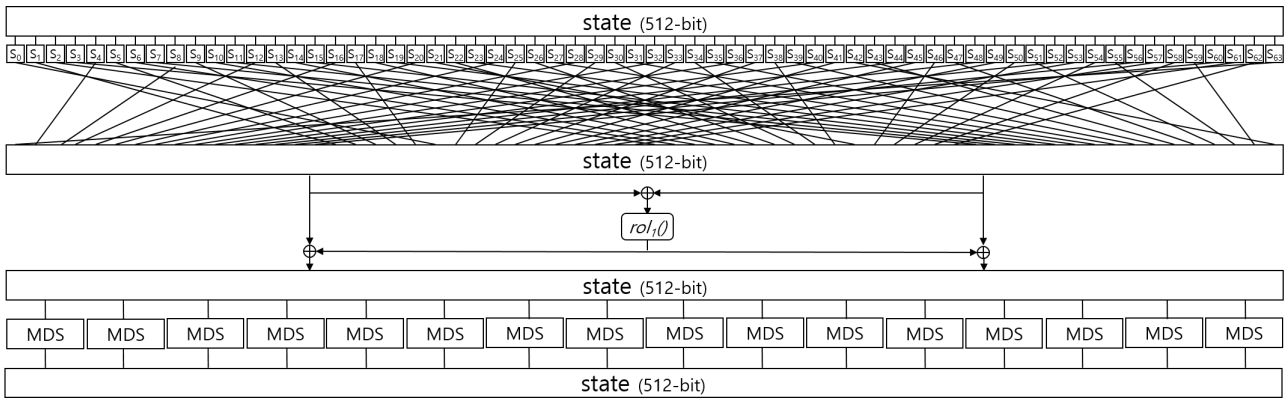

Figure 4.3: The schematic of round function

After the cryptanalysis, we believe that five/six/eight repetitions of the round function (when key length is 128/192/256-bit) are secure enough for our permutation P block presented in Figure 4.1.

4.3 Features/Functionalities

In this section, we illustrate and explain the implementation of the most required features of authenticated encryption.

Online and Inverse-free

The authenticated encryption scheme based on the duplex construction of the sponge function is naturally online and inverse-free. Figure 4.4 shows that the proposed scheme does not require an inverse function to decrypt ciphertext. Also, in Figure 4.4, the number of permutation block P_n is equal to the number of input blocks or the number of output blocks.

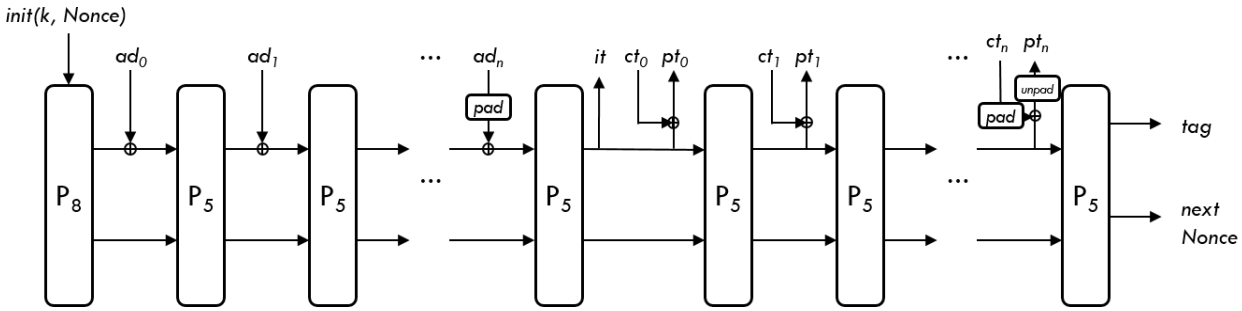

Figure 4.4: Decryption of the proposed scheme

Incremental AD/AE

We follow the approach from [8] to implement incremental AD/AE in our scheme. After one execution of the scheme, any few additional plaintext block or associated data block can be encrypted or authenticated without reinitialization of variables. In Figure 4.5, an example execution of incremental AE is illustrated.

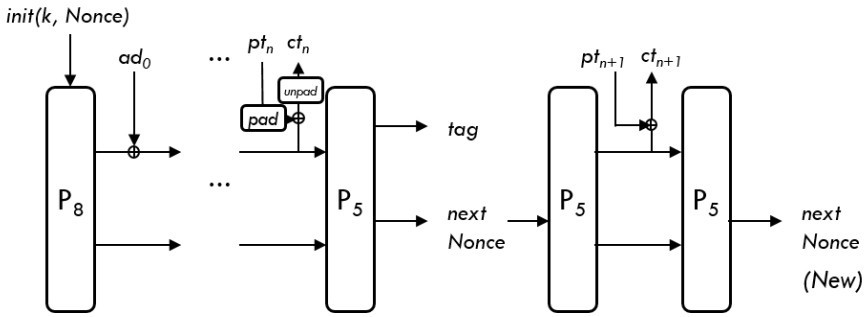

Figure 4.5: Incremental AE of the scheme

Intermediate tag, nonce-MR, and decryption-MR

The proposed authenticated encryption scheme generates intermediate tag it before encrypting plaintext, and the scheme generates $nextNonce$ for the subsequent execution of the scheme. After sharing $Nonce$ at the first execution, the scheme does not require $Nonce$. Thus, the scheme provides nonce-MR. At decryption, the calculated intermediate tag it will be tested with the transmitted it. If both values are different, the scheme refuses to proceed any further. Therefore, decryption will not be processed, and unverified plaintext will not be released.

No ciphertext expansion

The scheme obviously supports no ciphertext expansion feature. Using sponge-compliant multi-rate padding [8] and unpadding, the number of bits in plaintext is equal to the number of bits in ciphertext.

Parallelizability

The sponge function is inherently serial. However, any scheme based on the sponge function can be parallelized in the same way that other block ciphers are parallelized using counter mode. In Figure 4.6, the parallel variation of the proposed scheme is illustrated. An unique counter value is exclusive-ored with the result of initialization subroutine, and each lane generates intermediate values for it_i, tag_i, and $nextNonce_i$. After the execution is finished, the final values will be computed as follows: $it_f = \oplus_{i=0}^{\alpha}(it_i), tag_f = \oplus_{i=0}^{\alpha}(tag_i), nextNonce_f = \oplus_{i=0}^{\alpha}(nextNonce_i)$, where α is the number of lanes (parallelization depth).

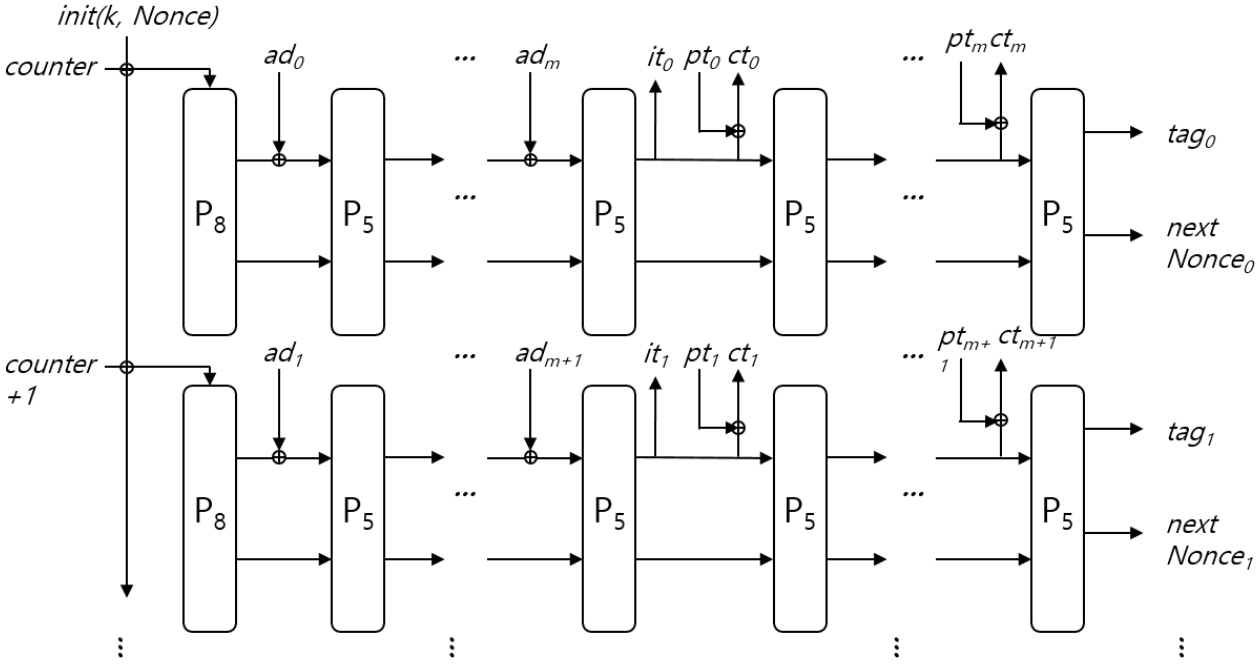

Figure 4.6: Parallel variation of the scheme

Chapter 5. Analysis

We have described the design of our novel lightweight authenticated encryption scheme based on the duplex construction of the sponge function. In this section, we analyze the performance and the security of the proposed scheme, and compare the scheme with other sponge-based authenticated encryption schemes submitted to the CAESAR competition.

5.1 Performance Analysis

We have implemented the proposed authenticated encryption scheme in software, The developing environment used is *bitset* template in C++ language and Microsoft Visual Studio 2012. AES-GCM is used as the basis of the performance evaluation of the authenticated encryption schemes submitted to the CAESAR competition. For the fair comparison, we implement AES-GCM using same C++ template. We measure the performance of our scheme and AES-GCM using *clock()* function. *clock()* function can be used to measure the wall-clock time of a program, and the unit is the number of clock ticks. We conduct the performance analysis using 8,400 byte, 4,800 byte, and 1,200 byte of input data. For each sizes of input data, we change the proportion of associated data and plaintext and calculate the average. The overall performance improvement is measured to be about 40%. The summary of our performance analysis is illustrated in In Table 5.1. The implementation using C++ *bitset* template gives an unexpected result. The performance bottleneck of AES-GCM should be GCM part, but the inefficiency of the *bitset* template makes AES to be the performance bottleneck of AES-GCM. At this time, we have used *bitset* template for better visualization of code and easier debugging. If our scheme and AES-GCM are implemented using *array* instead of *bitset* template, then we will get more realistic results.

The performances of other authenticated encryption schemes cannot be compared directly with the performance of our proposed scheme, because each authenticated encryption schemes provide their performance in *cpb* (cycles per byte) using different environment. The software performance of ICEPOLE-128 [18] is about 8 cycles per byte under Intel Xeon E3-1275 (Haswell). The software performances of

Ascon-128 [27] and NORX-128 [28] are less than 11 cycles per byte under Intel Sandy Bridge processor and NORX claims to have less than 7 cycles per byte under Intel Haswell processor (Intel Core i7-4770k at 3.5 GHz). However, we believe our proposed scheme will have a better performance-cost ratio compared to ICEPOLE, because we limit the size of width b and maximize the size of bitrate r as much as possible using [12].

In addition, we have not implemented the scheme in hardware. The hardware performance evaluation and comparison with other schemes will be done in future.

Table 5.1: Performance Evaluation

Total data (byte)	*pt* (byte)	*ad* (byte)	**AES-GCM (avg.)** (clock tick)	**Our scheme (avg.)** (clock tick)	**Improvement** (%)
8,400	8,000	400	2,274.3	1,300.7	74.9
8,400	6,400	2,000	1,825.7	1,299.7	40.5
8,400	4,800	3,600	1,378.7	1,310.3	5.2
4,800	4,560	240	1,290.0	749.3	72.2
4,800	3,600	1,200	1,026.7	742.3	38.3
4,800	2,640	2,160	754.0	740.0	1.9
1,200	1,136	64	317.3	187.0	69.7
1,200	896	304	225.0	187.0	36.4
1,200	656	544	182.0	177.0	2.8

5.2 Security Analysis

In this section, we present the security analysis of the proposed authenticated encryption scheme. Firstly, the provable security against the generic attacks is stated. Secondly, the resistance of the round function against various cryptanalysis techniques is proved.

5.2.1 Generic Attack

Like other authenticated encryption schemes based on the duplex construction of the sponge function, the proposed scheme inherits the general security claim from [8]. The duplex construction is a framework of the sponge function specialized for the authenticated encryption. The security bound of the sponge function is known to be $\min\{2^{c/2}, 2^k\}$, and the security bound of the duplex construction is known to be $\min\{2^{c-a}, 2^k\}$ [7]. However, Jovanovic *et al.* [12] proved that the security bound of authenticated encryption schemes based on the sponge function is $\min\{2^{b/2}, 2^c, 2^k\}$. Our proposed scheme inherits the security claim of [12] against the generic attacks, and IND-CPA and INT-CTXT security bounds are

stated in 2.2.2.

5.2.2 Cryptanalysis on Round Function

An authenticated encryption scheme based on the duplex construction of the sponge function is secure, if and only if the permutation block P_n is secure [8][18]. We analyze the security of our P_n block via some basic cryptanalysis techniques.

Differential and Linear Cryptanalysis

For differential and linear cryptanalyses, we require two information; the number of active S-boxes and the differential/linear characteristic/approximation. The AES S-box used in the scheme has differential/linear characteristic/approximation of $8/256 = 2^{-5}$. Furthermore, we believe that there are 8 active S-boxes in two consecutive rounds. Our BP layer is designed to diffuse an output of an S-box into two different S-boxes in subsequent round. Our bP layer, *latinSquare()*, is equivalent to a $(2, 2)$-multipermutation and our MP layer is equivalent to a $(4, 4)$-multipermutation. According to Mileva [17], changing one input value in an (ρ, γ)-multipermutation will change at least γ output values. Thus, we conclude that at least five/six/eight rounds are necessary for key parameter of 128/192/256-bit.

Rotational Cryptanalysis

For rotational cryptanalysis where "the adversary investigates the propagation of the rotational relations between the plaintexts" [18], we solve two weaknesses found in rotational cryptanalysis on Keccak. We follow the approach done by ICEPOLE [18]. According to [19], Keccak has two weak properties which are very low Hamming weight round constants and zero-initialized state. Our scheme does not use round constants at this stage, and we will add round constants with high Hamming weight if necessary. The proposed scheme initializes the state using a random number. Therefore, we believe that our authenticated encryption scheme can be secure against rotational cryptanalysis.

Techniques exploiting Low Algebraic Degree

For techniques exploiting low algebraic degree, we obtain the algebraic degree of one round of permutation block P_n. Again, we follow the approach done by ICEPOLE [18]. The algebraic degree of AES S-box is 7, so the algebraic degree will be 343 after three rounds. The algebraic degree of 343 prevents the algebraic attacks from meeting the attack complexity lower than the security level claim

of 2^{128}, 2^{192}, or 2^{256}. Therefore, the proposed authenticated encryption scheme is secure against the techniques exploiting low algebraic degree after 8-round initialization.

5.3 Feature Analysis

In this section, we compare the supportability of the most required authenticated encryption features between our scheme and other authenticated encryption schemes based on the sponge function. We extract the supportability of other schemes from Abed *et al.* [1], and present the comparison in Table 5.2. Our scheme supports the most features compared to any existing authenticated encryption schemes. The supported features of our proposed scheme are onlineness, inverse-freeness, parallelizable variation, incremental AD/AE, intermediate tag, provable security, nonce-MR, decryption-MR, and no ciphertext expansion.

Table 5.2: Comparison of features

Schemes	**Ascon**	**ICEPOLE**	**NORX**	**Our scheme**
Online	◯	◯	◯	◯
Inverse-free	◯	◯	◯	◯
Parallelizability	×	△	◯	△
Incremental AD/AE	×	×	×	◯
Intermediate tag	×	△	×	◯
Provable security	×	◯	◯	◯
Nonce-MR	◯	◯	×	◯
Decryption-MR	×	×	×	◯
No ciphertext expansion	◯	◯	◯	◯

◯ : support
△ : limited support
× : no support

Chapter 6. Concluding Remark

This thesis proposes a novel authenticated encryption scheme based on the duplex construction of the sponge function. The inputs of an authenticated encryption scheme are associated data, nonce, plaintext, and secret key, and the outputs are ciphertext and tag. In decryption, the inputs are associated data, nonce, ciphertext, secret key, and tag, and the outputs are plaintext and tag. If the transmitted tag does not equal to the calculated tag, the decryption algorithm outputs nothing. In addition, our proposed scheme generates intermediate tag and next nonce value for additional security. The scheme supports 128, 192, and 256-bit key sizes, and maximizes the size of bitrate to increase the performance. The round function consists of one nonlinear layer and three linear layers.

The authenticated encryption scheme provides provable security which is inherited from [8] and [12]. The security against generic attacks is proved under assumption that the underlying round function is secure. We prove that our round function has the resistance against various cryptanalysis techniques including differential cryptanalysis and linear cryptanalysis.

The performance of our scheme is compared with AES-GCM, an well-known authenticated encryption scheme. The evaluation result shows that our scheme is faster than AES-GCM by 40% on average. Our proposed scheme supports the most authenticated encryption features like onlineness, inverse-freeness, intermediate tag, nonce-MR, *etc.* than any existing authenticated encryption schemes.

Furthermore, future research is still remaining. Firstly, more cryptanalyses should be conducted on our round function for further security analysis. Secondly, the scheme and AES-GCM should be implemented without C++ *bitset* template for more realistic performance analysis. Lastly, the scheme should be implemented in hardware. Many authenticated encryption schemes submitted to the CAESAR competition [5] include the hardware implementation and performance analysis.

References

[1] Abed, F., Forler, C., and Lucks, S. (2014). "Classification of the CAESAR candidates", IACR Cryptology ePrint Archive, 2014.

[2] An J. H. and Bellare M. (2001). "Does encryption with redundancy provide authenticity?", Advances in Cryptology—EUROCRYPT 2001, Springer Berlin Heidelberg, pp. 512-528, 2001.

[3] Bellare M. and Namprempre C. (2000) "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm", Advances in Cryptology—ASIACRYPT 2000, Springer Berlin Heidelberg, pp. 531-545, 2000.

[4] Bellare M. and Rogaway P. (2001). "Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography", Advances in Cryptology—ASIACRYPT 2000, Springer Berlin Heidelberg, pp. 317-330, 2000.

[5] Bernstein, D. J. (2014) "CAESAR - competition for authenticated encryption: security, applicability, and robustness", 2014. [online]. Available at: http://competitions.cr.yp.to/caesar.html.

[6] Bertoni, G., Daemen, J., Peeters, M., and Assche, G. V. (2007). "Sponge functions", ECRYPT hash workshop, vol. 2007, 2007.

[7] Bertoni, G., Daemen, J., Peeters, M., and Assche, G. V. (2011). "On the security of the keyed sponge construction", SKEW 2011, 2011.

[8] Bertoni, G., Daemen, J., Peeters, M., and Assche, G. V. (2012). "Duplexing the sponge: single-pass authenticated encryption and other applications", Selected Areas in Cryptography, Springer Berlin Heidelberg, pp. 320-337 2012.

[9] Bogdanov, A. *et al.* (2007). "PRESENT: an ultra-lightweight block cipher", Springer Berlin Heidelberg, pp. 450-466, 2007.

[10] Dierks T. and Rescorla E. (2008) "The transport layer security (TLS) protocol version 1.2", RFC 5246, August 2008.

[11] Dworkin, M. (2007). "NIST special publication 800-38D", NIST special publication, vol. 800, pp. 38D, 2007.

[12] Jovanovic, P., Luykx, A., and Mennink, B. (2014). "Beyond $2^{c/2}$ security in sponge-based authenticated encryption modes", Advances in Cryptology–ASIACRYPT 2014, Springer Berlin Heidelberg, pp. 85-104, 2014.

[13] Junod, P. and Vaudenay, S. (2005). "Perfect diffusion primitives for block ciphers", Selected Areas in Cryptography, Springer Berlin Heidelberg, pp. 84-99, 2005.

[14] Junod, P. and Vaudenay, S. (2005). "FOX: a new family of block ciphers", Selected Areas in Cryptography, Springer Berlin Heidelberg, pp. 114-129, 2005.

[15] Kinney P. (2003). "Zigbee technology: wireless control that simply works", Communications design conference, vol. 2, 2003.

[16] Li, Y., and Wang, M. (2014). "Constructing s-boxes for lightweight cryptography with feistel structure", Cryptographic Hardware and Embedded Systems–CHES 2014, Springer Berlin Heidelberg, pp. 127-146, 2014.

[17] Mileva, A. (2014). "Multipermutations in crypto world: different faces of the perfect diffusion layer", Cryptology ePrint Archive, no. 85, 2014.

[18] Morawiecki, P. *et al.* (2014). "ICEPOLE: high-speed, hardware-oriented authenticated encryption", Cryptographic Hardware and Embedded Systems–CHES 2014, Springer Berlin Heidelberg, pp. 392-413, 2014.

[19] Morawiecki, P., Pieprzyk, J., and Srebrny, M. (2014). "Rotational cryptanalysis of round-reduced Keccak", Fast Software Encryption, Springer Berlin Heidelberg, pp. 241-262, 2014.

[20] Peralta, P. *et al.* (2015). "NIST randomness beacon", April 1st, 2015. [online]. Available at: http://www.nist.gov/itl/csd/ct/nist_beacon.cfm.

[21] Perlner, R. *et al.* (2012). "Third-round report of the SHA-3 cryptographic hash algorithm competition", US Department of Commerce, National Institute of Standards and Technology, 2012.

[22] Ristenpart, T. and Rogaway, P. (2007). “How to enrich the message space of a cipher”, Fast Software Encryption, Springer Berlin Heidelberg, pp. 101-118, 2007.

[23] Rogaway, P., Bellare, M., and Black, J. (2003). “OCB: A block-cipher mode of operation for efficient authenticated encryption”, ACM Transactions on Information and System Security (TISSEC), vol. 6, no. 3, pp. 365-403, 2003.

[24] Schnorr, C. P. and Vaudenay, S. (1994). “Parallel FFT-hashing”, Fast Software Encryption, Springer Berlin Heidelberg, 1994.

[25] Schnorr, C. P. and Vaudenay, S. (1995). “Black box cryptanalysis of hash networks based on multipermutations”, Advances in Cryptology—EUROCRYPT’94, Springer Berlin Heidelberg, 1995.

[26] Shannon, C. E. (1949). “Communication theory of secrecy systems”, Bell System Technical Journal, vol. 28, no. 4, 1949.

[27] Todo, Y., Sugawara, T., Murakami, Y., Aoki, K., and Matsui, M. (2015). “Design for involutive symmetric-key primitive”, The 32nd Symposium on Cryptography and Information Security–SCIS 2015, pp. 1D2-3, January 2015.

[28] Vaudenay, S. (1995). “On the need for multipermutations: cryptanalysis of MD4 and SAFER”, Fast Software Encryption, Springer Berlin Heidelberg, 1995.

[29] Whiting, D., Housley, R., and Ferguson, N. (2003). “Counter with CBC-MAC (CCM)”, RFC 3610, September 2003.

Appendices

A Pseudocode of the Scheme

```
//S is state, r is bitrate, c is capacity, k is key, pt is plaintext, ad is associated
    data
//it is intermediate tag

Procedure authenticatedEncryption (pt, ad, Nonce, k) :
  //initialize state
  init(S, Nonce, k)
  if(|k| == 128) then repetition = 5
  else if(|k| == 192) then repetition = 6
  else if(|k| == 256) then repetition = 8
  //padding associated data and plaintext
  ad = pad(ad)
  pt = pad(pt, |it|)
  //processing associated data
  foreach adi in ad
    roundFunction(S, repetition)
  //extracting intermediate tag
  it = truncate(r, |it|)
  //processing plaintext
  foreach pti in pt
    cti = XOR(r, pti)
    roundFunction(S, repetition)
  //unpadding last ciphertext block
  unpad(ctn)
  //generating tag
  tag = r
  //generating next Nonce
  next_Nonce = c
  return ct, Nonce, tag, it

Procedue authenticatedDncryption (ct, ad, tag, it, Nonce, k) :
  //initialize state
  init(S, Nonce, k)
  if(|k| == 128) then repetition = 5
  else if(|k| == 192) then repetition = 6
  else if(|k| == 256) then repetition = 8
  //padding associated data and ciphertext
  ad = pad(ad)
  ct = pad(ct, |it|)
  //processing associated data
  foreach adi in ad
    roundFunction(S, repetition)
  //extracting temporary intermediate tag
  temp_it = truncate(r, |it|)
  //testing intermediate tag
  if temp_it != it then return NULL
  //processing ciphertext
```

```
foreach cti in ct
  pti = XOR(r, cti)
  roundFunction(S, repetition)
//unpadding last plaintext block
unpad(ptn)
//generating temporary tag
temp_tag = r
//generating next Nonce
next_Nonce = c
//testing tag
if temp_tag == tag then return pt
else return NULL
```

Pseudocode

B Source code of the Scheme

```
#include "stdafx.h"
#include <bitset>
#include <iostream>
#include <time.h>

using namespace std;

//List of Tables
int pBox[64] = {0, 16, 32, 48, 1, 17, 33, 49, 2, 18, 34, 50, 3, 19, 35, 51, 4, 20, 36,
    52, 5, 21, 37, 53, 6, 22, 38, 54, 7, 23, 39, 55, 8, 24, 40, 56, 9, 25, 41, 57, 10,
    26, 42, 58, 11, 27, 43, 59, 12, 28, 44, 60, 13, 29, 45, 61, 14, 30, 46, 62, 15, 31,
    47, 63};

int mds[16] = {0x01, 0x01, 0xfd, 0x02, 0x01, 0xfd, 0x02, 0x01, 0x01, 0x02, 0x01, 0xfd, 0
    x02, 0x01,0x01,0x01};

//AES Sbox
int sbox[16] = {0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30, 0x01, 0x67, 0x2B,
    0xFE, 0xD7, 0xAB, 0x76, 0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4,
    0xA2, 0xAF, 0x9C, 0xA4, 0x72, 0xC0, 0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC,
    0x34, 0xA5, 0xE5, 0xF1, 0x71, 0xD8, 0x31, 0x15, 0x04, 0xC7, 0x23, 0xC3, 0x18, 0x96,
    0x05, 0x9A, 0x07, 0x12, 0x80, 0xE2, 0xEB, 0x27, 0xB2, 0x75, 0x09, 0x83, 0x2C, 0x1A,
    0x1B, 0x6E, 0x5A, 0xA0, 0x52, 0x3B, 0xD6, 0xB3, 0x29, 0xE3, 0x2F, 0x84, 0x53, 0xD1,
    0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B, 0x6A, 0xCB, 0xBE, 0x39, 0x4A, 0x4C, 0x58, 0xCF,
    0xD0, 0xEF, 0xAA, 0xFB, 0x43, 0x4D, 0x33, 0x85, 0x45, 0xF9, 0x02, 0x7F, 0x50, 0x3C,
    0x9F, 0xA8, 0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D, 0x38, 0xF5, 0xBC, 0xB6, 0xDA, 0x21,
    0x10, 0xFF, 0xF3, 0xD2, 0xCD, 0x0C, 0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17, 0xC4, 0xA7,
    0x7E, 0x3D, 0x64, 0x5D, 0x19, 0x73, 0x60, 0x81, 0x4F, 0xDC, 0x22, 0x2A, 0x90, 0x88,
    0x46, 0xEE, 0xB8, 0x14, 0xDE, 0x5E, 0x0B, 0xDB, 0xE0, 0x32, 0x3A, 0x0A, 0x49, 0x06,
    0x24, 0x5C, 0xC2, 0xD3, 0xAC, 0x62, 0x91, 0x95, 0xE4, 0x79, 0xE7, 0xC8, 0x37, 0x6D,
    0x8D, 0xD5, 0x4E, 0xA9, 0x6C, 0x56, 0xF4, 0xEA, 0x65, 0x7A, 0xAE, 0x08, 0xBA, 0x78,
    0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6, 0xE8, 0xDD, 0x74, 0x1F, 0x4B, 0xBD, 0x8B, 0x8A,
    0x70, 0x3E, 0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E, 0x61, 0x35, 0x57, 0xB9, 0x86, 0xC1,
    0x1D, 0x9E, 0xE1, 0xF8, 0x98, 0x11, 0x69, 0xD9, 0x8E, 0x94, 0x9B, 0x1E, 0x87, 0xE9,
    0xCE, 0x55, 0x28, 0xDF, 0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68, 0x41, 0x99,
    0x2D, 0x0F, 0xB0, 0x54, 0xBB, 0x16};

int mBox_2[256] = {0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34,
    36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76,
    78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100, 102, 104, 106, 108, 110, 112, 114,
    116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146, 148,
     150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180,
    182, 184, 186, 188, 190, 192, 194, 196, 198, 200, 202, 204, 206, 208, 210, 212, 214,
     216, 218, 220, 222, 224, 226, 228, 230, 232, 234, 236, 238, 240, 242, 244, 246,
    248, 250, 252, 254, 27, 25, 31, 29, 19, 17, 23, 21, 11, 9, 15, 13, 3, 1, 7, 5, 59,
    57, 63, 61, 51, 49, 55, 53, 43, 41, 47, 45, 35, 33, 39, 37, 91, 89, 95, 93, 83, 81,
    87, 85, 75, 73, 79, 77, 67, 65, 71, 69, 123, 121, 127, 125, 115, 113, 119, 117, 107,
     105, 111, 109, 99, 97, 103, 101, 155, 153, 159, 157, 147, 145, 151, 149, 139, 137,
    143, 141, 131, 129, 135, 133, 187, 185, 191, 189, 179, 177, 183, 181, 171, 169, 175,
     173, 163, 161, 167, 165, 219, 217, 223, 221, 211, 209, 215, 213, 203, 201, 207,
    205, 195, 193, 199, 197, 251, 249, 255, 253, 243, 241, 247, 245, 235, 233, 239, 237,
     227, 225, 231, 229};
```

```
int mBox_fd[256] = {0, 253, 225, 28, 217, 36, 56, 197, 169, 84, 72, 181, 112, 141, 145,
    108, 73, 180, 168, 85, 144, 109, 113, 140, 224, 29, 1, 252, 57, 196, 216, 37, 146,
    111, 115, 142, 75, 182, 170, 87, 59, 198, 218, 39, 226, 31, 3, 254, 219, 38, 58,
    199, 2, 255, 227, 30, 114, 143, 147, 110, 171, 86, 74, 183, 63, 194, 222, 35, 230,
    27, 7, 250, 150, 107, 119, 138, 79, 178, 174, 83, 118, 139, 151, 106, 175, 82, 78,
    179, 223, 34, 62, 195, 6, 251, 231, 26, 173, 80, 76, 177, 116, 137, 149, 104, 4,
    249, 229, 24, 221, 32, 60, 193, 228, 25, 5, 248, 61, 192, 220, 33, 77, 176, 172, 81,
     148, 105, 117, 136, 126, 131, 159, 98, 167, 90, 70, 187, 215, 42, 54, 203, 14, 243,
     239, 18, 55, 202, 214, 43, 238, 19, 15, 242, 158, 99, 127, 130, 71, 186, 166, 91,
    236, 17, 13, 240, 53, 200, 212, 41, 69, 184, 164, 89, 156, 97, 125, 128, 165, 88,
    68, 185, 124, 129, 157, 96, 12, 241, 237, 16, 213, 40, 52, 201, 65, 188, 160, 93,
    152, 101, 121, 132, 232, 21, 9, 244, 49, 204, 208, 45, 8, 245, 233, 20, 209, 44, 48,
     205, 161, 92, 64, 189, 120, 133, 153, 100, 211, 46, 50, 207, 10, 247, 235, 22, 122,
     135, 155, 102, 163, 94, 66, 191, 154, 103, 123, 134, 67, 190, 162, 95, 51, 206,
    210, 47, 234, 23, 11, 246};

//left-circular rotation
bitset<256> rol(bitset<256> state){
  bool a = state[state.size()-1];
  state = state <<= 1;
  state[0] = a;
  return state;
}

//nonlinear layer
bitset<8> sBox(bitset<8> a){
  unsigned long a_i = a.to_ulong();
  a_i = sbox[a_i];
  a.reset();
  bitset<8> ta(a_i);
  for(int i = 15; i >= 0; i--){
    a[i] = ta[i];
  }
  return a;
}

//apply Sbox to 512-bit state
bitset<512> sBox_512(bitset<512> state){
  bitset<8> temp;
  for(int i = 63; i >= 0; i--){
    int k = (i+1)*8-1;
    for(int j = 7; j >= 0; j--){
      temp[j] = state[k--];
    }
    k = k + 8;
    temp = sBox(temp);
    for(int j = 7; j >= 0; j--){
      state[k--] = temp[j];
    }
  }
  return state;
}

//latin square
bitset<512> latinSquare(bitset<512> state){
```

```
  bitset<256> left, right, ex_t;
  for(int i = state.size()-1; i >= state.size()/2; i--){
    left[i - left.size()] = state[i];
  }
  for(int i = state.size()/2-1; i >= 0; i--){
    right[i] = state[i];
  }
  ex_t = rol(left ^ right);
  left ^= ex_t;
  right ^= ex_t;
  for(int i = state.size()-1; i >= state.size()/2; i--){
    state[i] = left[i - left.size()];
  }
  for(int i = state.size()/2-1; i >= 0; i--){
    state[i] = right[i];
  }
  return state;
}

//modified PRESENT permutation layer
bitset<512> bytePerm(bitset<512> state){
  bool a = state[state.size()-1];
  bool b = state[state.size()-2];
  state = state <<= 2;
  state[1] = a;
  state[0] = b;
  bitset<512> temp = state;
  int sent, t_i;
  for(int i = 0; i < state.size()/8; i++){
    sent = (pBox[i]+1)*8-1;
    t_i = (i+1)*8-1;
    state[sent--] = temp[t_i--];
    state[sent--] = temp[t_i--];
    state[sent--] = temp[t_i--];
    state[sent--] = temp[t_i--];
    state[sent--] = temp[t_i--];
    state[sent--] = temp[t_i--];
    state[sent--] = temp[t_i--];
    state[sent--] = temp[t_i--];
  }
  return state;
}

//MDS (each byte)
bitset<8> matrixMult_subbyte(bitset<8> t, int index){
  unsigned long t_i = t.to_ulong();
  if(mds[index] == 0x01)
    return t;
  else if(mds[index] == 0x02){
    bitset<8> temp(mBox_2[t_i]);
    return temp;
  }
  else if(mds[index] == 0xfd){
    bitset<8> temp(mBox_fd[t_i]);
    return temp;
```

```
  }
  return t;
}

//MDS (4 bytes)
bitset<8> matrixMult_subrow(bitset<8> t1, bitset<8> t2, bitset<8> t3, bitset<8> t4, int
    index){
  bitset<8> result(0);
  result ^= matrixMult_subbyte(t1, index++);
  result ^= matrixMult_subbyte(t2, index++);
  result ^= matrixMult_subbyte(t3, index++);
  result ^= matrixMult_subbyte(t4, index);
  return result;
}

//MDS (16 bytes)
bitset<32> matrixMult(bitset<32> m){
  bitset<8> t1, t2, t3, t4, r1, r2, r3, r4;
  for(int i = 31; i >=0; i--){
    if(i > 23)
      t1[i - 24] = m[i];
    else if(i > 15)
      t2[i - 16] = m[i];
    else if(i > 7)
      t3[i - 8] = m[i];
    else
      t4[i] = m[i];
  }
  r1 = matrixMult_subrow(t1, t2, t3, t4, 0);
  r2 = matrixMult_subrow(t1, t2, t3, t4, 4);
  r3 = matrixMult_subrow(t1, t2, t3, t4, 8);
  r4 = matrixMult_subrow(t1, t2, t3, t4, 12);
  for(int i = 31; i >=0; i--){
    if(i > 23)
      m[i] = r1[i - 24];
    else if(i > 15)
      m[i] = r2[i - 16];
    else if(i > 7)
      m[i] = r3[i - 8];
    else
      m[i] = r4[i];
  }
  return m;
}

//MDS for whole state
bitset<512> matrixPerm(bitset<512> state){
  bitset<32> temp;
  for(int i = (state.size()/32-1); i >= 0; i--){
    int a = (i + 1) * 32 - 1;
    for(int j = 31; j >= 0; j--){
      temp[j] = state[a--];
    }
    temp = matrixMult(temp);
    a += 32;
```

```
    for(int j = 31; j >=0; j--){
      state[a--] = temp[j];
    }
    temp.reset();
  }
  return state;
}

//field multiplication code for table generation
unsigned long field_mult(unsigned long m){
  bitset<16> result(0);
  bitset<16> temp(0);
  bitset<8> m1(m);
  bitset<8> m2(0xfd);
  bitset<9> rp(0x11b);
  for(int j = 7; j >= 0; j--){
    if(m1[j] == 1){
      for(int i = 7; i >= 0; i--){
        if(m2[i] == 1){
          temp[j+i] = 1;
        }
      }
      result ^= temp;
      temp.reset();
    }
  }
  for(int i = 15; i >= 8; i--){
    if(result[i] == 1){
      for(int j = 8; j >= 0; j--){
        result[i+j-8] = result[i+j-8] ^ rp[j];
      }
    }
  }
  if(result.to_ulong() >= 256){
    cout << "ERROR" << endl;
  }
  return result.to_ulong();
}


//round function
bitset<512> roundFunc(bitset<512> state, int iteration){
  for(int i = 0; i < iteration; i++){
    //sbox
    state = sBox_512(state);
    //bytePerm
    state = bytePerm(state);
    //bitPerm
    state = latinSquare(state);
    //MatrixPerm
    state = matrixPerm(state);
  }
  return state;
}
```

```
//associated data processing
bitset<512> processAD_128(bitset<512> state, bitset<1920> ad){
  bitset<128> it;
  state = roundFunc(state, 8);
  for(int i = ad.size() / (state.size() - 128); i > 0; i--){
    int k = ad.size() / i - 1;
    for(int l = 511; l >= it.size(); l--){
      state[l] = state[l] ^ ad[k--];
    }
    state = roundFunc(state, 5);
  }
  int j = it.size()-1;
  for(int i = 511; i >= state.size()-it.size(); i--){
    it[j--] = state[i];
  }
  return state;
}

//plaintext processing
bitset<512> processAE_128(bitset<512> state, bitset<31104> pt){
  bitset<384> ct;
  for(int i = 1; i <= pt.size() / (state.size() - 128); i++){
    int k = pt.size() / i - 1;
    int m = 384 - 1;
    for(int l = 511; l >= 128; l--){
      state[l] = state[l] ^ pt[k--];
      ct[m--] = state[l];
    }
    state = roundFunc(state, 5);
  }
  return state;
}

//authenticated encryption
bitset<512> authEnc_128(bitset<512> state, bitset<1920> ad, bitset<31104> pt, bitset
    <128> key, bitset<384> nonce){
  bitset<128> tag;
  bitset<128> it;
  int ki = 127;
  int ni = 383;
  bitset<512> init_state(0x0x20AAF8DDB4E40B2814B93E0B57BBB336777DAF8B60BACF06A131EB7
                E0D8C923CD184E3A1C1031A3B3AFBC89601CAD91CC28C6DA918A7E4E
                37DB46EED18A9607C);
  state = init_state;
  for(int i = 511; i >= 0; i--){
    if(i >= 384)
      state[i] = key[ki--];
    else if(i < 384)
      state[i] = nonce[ni--];
  }
  state = latinSquare(state);
  state = processAD_128(state, ad);
  for(int i = 511; i>= 512-128; i--){
    it[i] = state[i]
  }
```

```
  state = processAE_128(state, pt);
  int j = 127;
  for(int i = 511; i >= 384; i--){
    tag[j--] = state[i];
  }
  return state;
}

//authenticated decryption
bitset<512> authDec_128(bitset<512> state, bitset<1920> ad, bitset<31104> ct, bitset
    <128> key, bitset<128> it, bitset<384> nonce, bitset<128> tag){
  bitset<128> temp_tag(0);
  int ki = 127;
  int ni = 383;
  bitset<512> init_state(0x0x20AAF8DDB4E40B2814B93E0B57BBB336777DAF8B60BACF06A131EB7
                E0D8C923CD184E3A1C1031A3B3AFBC89601CAD91CC28C6DA918A7E4E
                37DB46EED18A9607C);
  state = init_state;
  for(int i = 511; i >= 0; i--){
    if(i >= 384)
      state[i] = key[ki--];
    else if(i < 384)
      state[i] = nonce[ni--];
  }
  state = latinSquare(state);
  state = processAD_128(state, ad);
  bitset<128> temp_it(0);
  for(int i = 511; i>= 512-128; i--){
    temp_it[i] = state[i]
  }
  if(temp_it != it) return NULL;
  state = processAE_128(state, pt);
  int j = 127;
  for(int i = 511; i >= 384; i--){
    temp_tag[j--] = state[i];
  }
  if(temp_tag != tag) return NULL;
  return state;
}
```

Full source code

Summary

Design and Analysis of Authenticated Encryption Scheme based on the Sponge Function

인증암호화(Authenticated Encryption, AE)는 데이터의 암호화와 무결성 검증 및 상대방 인증을 동시에 수행하여 기밀성과 무결성 및 인증성을 제공하는 대칭키 암호 시스템이다. 일반적인 대칭키 암호 시스템은 데이터의 암호화를 통해 기밀성만 제공하고 일반적인 해쉬 함수는 무결성 혹은 인증성만을 제공한다.

다양한 정보보안의 요소 중 대표적으로 기밀성과 무결성 및 가용성이 있으며, IT 시스템을 안전하게 보호하기 위해서 시스템에 최대한 많은 정보보안의 요소들을 제공하는 것이 필수이다. 각 요소들을 제공하는 암호학적 프리미티브들은 모두 IT 시스템의 자원을 소모한다. 그러므로 기밀성과 무결성 및 인증성을 동시에 제공하는 인증암호화 기법은 자원을 효율적으로 사용하여 시스템을 공격으로부터 안전하게 보호할 수 있다. 또한, IoT 시대가 도래하면서 가용 자원이 매우 제한적인 작은 디바이스들도 인터넷에 연결되었기 때문에 자원을 효율적으로 사용하면서 시스템을 공격으로부터 보호하는 것이 더욱 중요하다.

표준 해쉬 함수인 SHA-3이 채택된 Keccak 함수는 Sponge function을 기반으로 한다. 그러나 Sponge function은 해쉬 함수 뿐만 아니라 Stream cipher와 인증암호화 등에도 사용될 수 있다. 인증암호화를 위한 Sponge function의 형태는 duplex construction 이라고 불리운다. Sponge function에 기반하는 인증암호화 기법은 암호학적 안전성이 검증되었고 구현 비용과 실행 속도에 강점을 지니며 다양한 기능을 기본적으로 제공한다. 하지만 Sponge function은 기본적으로 직렬적인 형태를 지니고 있어 데이터의 병렬 처리가 힘든 단점이 있다.

따라서, 본 논문에서는 Sponge function의 duplex construction에 기반한 인증암호화 기법을 제안한다. 제안하는 기법은 전체 설계 및 내부 라운드 함수에 대한 암호학적 안전성이 검증되었고 기존 표준인 AES-GCM보다 실행 속도가 빠르며 인증암호화의 속도와 안전성에 도움이 되는 다양한 기능들을 탑재하고 있다. 본 기법이 현존하는 인증암호화 기법 중 최다의 기능들을 제공하며 Sponge function의 병렬 형태도 지원하고 있다.

핵심어: Authenticated Encryption, Sponge Function, Symmetric Key Cryptography, Hash Function, CAESAR

감 사 의 글

이 논문을 작성하기까지 많은 분들께 다양한 도움을 받았습니다. 지도교수님이시자 제 멘토이신 김광조 교수님께는 적게는 연구의 진행 방향 뿐만 아니라, 연구자로써의 마음가짐과 태도에 이르기까지 많은 것을 배울 수 있어 제 인생의 이정표를 세우는데 정말로 큰 도움을 주셨습니다. 또한, 바쁘신 와중에도 학위논문심사에 참여하셔서 진심어린 조언을 주신 김대영 교수님, 김순태 교수님께도 감사의 말씀을 드립니다.

그리고 연구실의 동기, 선후배인 Khalid Telman Huseynov, 최락용 형님, 박준정 형님, 정제성 형님, 이동수, 안수현, 김경민, 홍정아, 김세호, Aminanto Erza Muhamad 에게도 연구 및 과제, 연구실 생활과 같은 부분에서 이야기를 나누면서 많은 격려를 받았습니다. 특히, 수학 문제에 있어 많은 도움을 준 최락용에게는 진심으로 감사드립니다.

이외에도 감사를 드릴 분들이 많습니다. 각종 연구실 업무에 도움을 주신 홍지연씨, 제 인생을 살아감에 있어 사회생활의 멘토가 되어주신 김정민 형님, 타 학과이지만 연구 태도 등 많은 도움을 주신 김대옥 형님, 김대우 형님께도 감사를 드립니다.

끝으로, 제가 언급하지 않은 모든 분들에게도 감사를 드립니다. 앞으로도 지금의 저보다 한층 더 나아진 모습으로 보답하겠습니다. 감사합니다.

이 력 서

이 름 : 김 학 주

생 년 월 일 : 1988년 1월 11일

E-mail 주 소 : ndemian@kaist.ac.kr

학 력

2002. 8. – 2006. 5. Jakarta International School

2006. 9. – 2013. 8. 한국과학기술원 전산학부 (B.S.)

2013. 9. – 2015. 8. 한국과학기술원 전산학부 (M.S.)

경 력

2012. 3. – 2012. 8. 한국과학기술원 프로그래밍기초 일반조교

2012. 9. – 2012. 12. 한국과학기술원 프로그래밍기초 일반조교

2013. 3. – 2013. 8. 한국과학기술원 프로그래밍기초 일반조교

2014. 3. – 2014. 8. 한국과학기술원 정보보호개론 일반조교

2014. 7. – 2014. 8. UC Berkeley 학부생 멘토

2014. 7. – 2014. 9. CHES 2014 운영 보조

2014. 9. – 2014. 12. 한국과학기술원 고급정보보호 일반조교

2015. 3. – 2015. 8. 한국과학기술원 프로그래밍기초 일반조교

2015. 3. – 2015. 8. 대전과학고등학교 심화자율연구 조교

연 구 과 제

2013. 1. – 2013. 10. Securing SCADA Protocols for Nuclear Plants

2013. 5. – 2013. 12. Intrusion Detection System for Critical Infrastructures

2013. 4. – 2015. 8. 생체모방 알고리즘(Bio-inspired Algorithm)을 활용한 통신기술 연구

2014. 7. – 2015. 6. Intrusion Detection System for Critical Infrastructures Using Big Data Analytics

2015. 4. – 2015. 8. 국가재난안전통신망의 장기간 보안성을 보장하는 산업 발전 전략

2015. 4. – 2015. 8. 포스트 양자암호 시스템의 안전성 분석기술 열구

연 구 업 적

1. **HakJu Kim** and Kwangjo Kim, "Authenticated Encryption for DNP3 in SCADA system", A3 Foresight Program 2013, 2013.07.11. Sapporo, Japan. - [Best Presentation Award]
2. **HakJu Kim** and Kwangjo Kim, "Empirical Approach to Enhance the Security of DNP3 Protocol in SCADA System using Low-latency Block Cipher", 2013 정보보호학술발표회논문집 충청지부, pp.72-84, 2013.09.27. 순천향대학교, 천안.
3. **HakJu Kim** and Kwangjo Kim, "Toward an Inverse-free Lightweight Encryption Scheme for IoT", 한국정보보호학회 동계 학술대회(CISC-W14), 2014.12.06. 한양대학교, 서울.
4. 정제성, 김경민, **김학주**, 박준정, 안수현, 이동수, 최락용, 김광조, 김대영, "경량 암호를 이용한 IoT Secure SNAIL 플랫폼 구성(I)", 한국정보보호학회 동계학술대회 (CISC-W'14), 2014.12.06. 한양대학교, 서울.
5. **HakJu Kim** and Kwangjo Kim, "Who can survive in CAESAR competition at round-zero?", 2014 Symposium on Cryptography and Information Security (SCIS 2014), 2014.01.21-24, Kagoshima, Japan.
6. Dongsoo Lee, **HakJu Kim**, Kwangjo Kim, and Paul D. Yoo, "Simulated Attack on DNP3 Protocol in SCADA System", 2014 Symposium on Cryptography and Information Security (SCIS 2014), 2014.01.21-24, Kagoshima, Japan.

7. **HakJu Kim**, Kwangjo Kim, “Preliminary Design of a Novel Authenticated Encryption Scheme based on the Sponge Function”, 10th Asia Joint Conference on Information Security (AsiaJCIS 2015), 2015.05.24-26. Kaohsiung, Taiwan.

8. 김광조, **김학주**, 후세이너브 칼리드, 홍지연, “P2P Botnet 탐지를 위한 적응가변형 침입 탐지 시스템의 방식 및 장치”, 출원번호 10-2015-0021688, 출원일 2015.02.12.