ตัวก่อกำเนิดบิตสุ่มเทียมสำหรับวิทยาการรหัสลับความเร็วสูง

นายวศิน สุทธิฉายา

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรดุษฎีบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2554

PSEUDO-RANDOM BIT GENERATOR FOR HIGH-SPEED CRYPTOGRAPHY

Mister Vasin Suttichaya

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy Program in Computer Science
Department of Mathematics and Computer Science
Faculty of Science
Chulalongkorn University
Academic year 2011

Thesis Title	PSEUDO-RANDOM BIT GENERATOR FOR HIGH-SPEED CRYPTOGRAPHY
By	Mister Vasin Suttichaya
Field of Study	Computer Science
Thesis Advisor	Assistant Professor Pattarasinee Bhattarakosol, Ph.D.
Thesis Co-advisor	Associate Professor Soraj Hongladarom, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Doctoral Degree

……Dean of the Faculty of Science
(Professor Supot Hannongbua, Dr.rer.nat.)

THESIS COMMITTEE

……Chairman
(Assistant Professor Nagul Cooharojananone, Ph.D.)

……Thesis Advisor
(Assistant Professor Pattarasinee Bhattarakosol, Ph.D.)

……Thesis Co-Advisor
(Associate Professor Soraj Hongladarom, Ph.D.)

……Examiner
(Assistant Professor Rajalida Lipikorn, Ph.D.)

……External Examiner
(Assistant Professor Ohm Sornil, Ph.D.)

……External Examiner
(Assistant Professor Benchaphon Limthanmaphon, Ph.D.)

วศิน สุทธิฉายา : ตัวก่อกำเนิดบิตสุ่มเทียมสำหรับวิทยาการรหัสลับความเร็วสูง. (PSEUDO-RANDOM BIT GENERATOR FOR HIGH-SPEED CRYPTOGRAPHY) อ. ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ. ดร.ภัทรสินี ภัทรโกศล, อ. ที่ปรึกษาวิทยานิพนธ์ร่วม : รศ.ดร. โสรัจจ์ หงศ์ลดารมภ์, 75 หน้า.

ตัวก่อกำเนิดบิตสุ่มเทียมเป็นขั้นตอนวิธีสำหรับสร้างเลขสุ่มเทียมจำนวนมากจากเลขสุ่มที่แท้จริงจำนวนน้อย เลขสุ่มเทียมที่ปลอดภัยจะต้องไม่มีขั้นตอนวิธีที่มีประสิทธิภาพสำหรับจำแนกเลขสุ่มเทียมออกจากเลขสุ่มที่แท้จริง เลขสุ่มเทียมถูกนำไปใช้ในงานด้านต่างๆ ของวิทยาการรหัสลับ เช่น ใช้เป็นกุญแจสำหรับการเข้ารหัสเป็นต้น ในงานวิจัยนี้ได้เสนอการวิเคราะห์ทางทฤษฎีของวิธีการโจมตีการเข้ารหัสแบบสตรีมไซเฟอร์ และการสร้างเครื่องสร้างเลขสุ่มเทียมที่มีประสิทธิภาพและความปลอดภัยมากขึ้นการโจมตีสตรีมไซเฟอร์ที่นำเสนอในงานวิจัยนี้มีข้อได้เปรียบเหนือการโจมตีทั่วไปในด้านความสามารถในการระบุจำนวนข้อมูลที่ต้องใช้ในการโจมตีและความถูกต้องในการทำงานเมื่อทรัพยากรมีจำกัด งานวิจัยนี้ได้นำเสนอตัวก่อกำเนิดบิตสุ่มเทียมแบบผสมผสานกับอัลกอริทึมเข้ารหัสแบบสตรีมไซเฟอร์เพื่อให้ได้ความปลอดภัยและประสิทธิภาพที่มากขึ้น งานวิจัยนี้ได้เสนอตัวก่อกำเนิดบิตสุ่มเทียมที่มีความปลอดภัยยิ่งยวดและสามารถต้านทานการโจมตีจากเทคโนโลยีควอนตัมคอมพิวเตอร์ ซึ่งความปลอดภัยดังกล่าวเหนือกว่าตัวก่อกำเนิดบิตสุ่มเทียมที่ใช้อยู่ในปัจจุบันมาก และยังได้ศึกษาความปลอดภัยของสมมติฐานทางการเข้ารหัสเมื่อข้อมูลบางส่วนของกุญแจสำหรับเข้ารหัสถูกล่วงรู้โดยผู้โจมตี

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์ ลายมือชื่อนิสิต.....

สาขาวิชา วิทยาการคอมพิวเตอร์ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก .....

ปีการศึกษา 2554 ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์ร่วม .....

5073869523 : MAJOR COMPUTER SCIENCE
KEYWORDS : PSEUDO-RANDOM GENERATOR / STREAM CIPHER / DISTINGUISHING ATTACK / LEARNING WITH ERROR / LEARNING PARITY WITH NOISE

VASIN SUTTICHAYA : PSEUDO-RANDOM BIT GENERATOR FOR HIGH-SPEED CRYPTOGRAPHY. ADVISOR : ASST.PROF. PATTARASINEE BHATTARAKOSOL, PH.D., CO-ADVISOR : ASSOC.PROF.SORAJ HONGLADAROM, PH.D., 75 pp.

Pseudorandom generator is the important cryptographic primitive. It expands the real short random sequence to the longer sequence that is hard to predict. These random sequences from pseudorandom generator are used as the secret keys, salts, nonces, and so on. It is impossible to construct the cryptosystems that do not use any random numbers. This thesis mainly devotes to theoretical analysis on pseudorandom generators and stream ciphers. The theoretical analysis on the distinguishing attack is also involved in this dissertation. The proposed distinguishing attack is based on binomial hypothesis testing, which is differ from the traditional distinguishing attack. This approach has an advantage when the memory is limited. The hybrid method between pseudorandom generator and stream cipher is proposed. The hybrid generator obtains high performance over the prototype pseudorandom generator. Moreover, the security proof shows that the hybrid generator also gains high security over the prototype stream cipher. This research also proposes the construction of extremely secure pseudorandom generator. The proposed pseudorandom generator is based on the hardness assumption of Learning with Error problem. The proof shows that the problem of distinguishing the random outputs from the proposed generator is at least as hard as solving Learning with Error problem, which cannot efficiently solve by any quantum computers. This research also works on theoretical analysis of the robustness of Learning Parity with Noise assumption when the secret information is leaked. The theoretical analysis shows that Learning Parity with Noise remains secure in the leaky-key environment but the security parameter is reduced to the min-entropy that is left in the secret key. The robustness property of Learning Parity with Noise assumption can be used to construct the leakage-resilient cryptosystems.

Department : Mathematics and Computer Science	Student's Signature .....
Field of Study : Computer Science	Advisor's Signature .....
Academic Year : 2011	Co-advisor's Signature .....

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study.

I would like to thank the Office of Higher Education Commission, and Chulalongkorn University for their financial support.

I would like to express my deepest appreciations to my advisor, Assist.Prof.Dr. Pattarasinee Bhattarakosol. I am greatly indebted to her for giving me the guidance and the morale. All of her advices help me conquer all difficulties and make this dissertation possible.

I would like to thank my co-advisor, Assoc.Prof.Dr. Soraj Hongladarom for his supportive and generous.

I also would like to thank the Department of Electrical Engineering at Katholieke Universiteit Leuven for their facility support during my visiting scholar in 2011. Additionally, I would like to give my appreciation to Prof.Dr. Bart Preneel for giving me an opportunity to work at COSIC and many useful comments on my research.

I am honored to have Assist.Prof.Dr. Nagul Cooharojananone, Assist.Prof.Dr. Rajalida Lipikorn, Assist.Prof.Dr. Ohm Sornil, and Assist.Prof.Dr. Benchaphon Limthanmaphon serve as my dissertation committees.

Finally, I would like to dedicate my Ph.D. to my parent who have given me the opportunity of an education from the best institutions and support throughout my life.

CONTENTS

Page

List of Tables

List of Figures

CHAPTER I

INTRODUCTION

This dissertation concerns about the theoretical research of randomness in cryptography. The main topics in this thesis consist of the construction of provable secure pseudorandom generator, the distinguishing attack on the stream ciphers, the trade-off between security and performance, and the hardness of cryptographic assumption when the secret information is leaked. In this chapter, the motivations and objectives of this research are presented.

This chapter is organized into 6 sections. In Section 1.1, the problems and motivations of this dissertation are described. The objectives of this thesis are listed in Section 1.2. The scopes and limitations are determined in Section 1.3. The main contributions of this thesis are presented in Section 1.4. The research plan is shown in Section 1.5. Finally, the organization of this dissertation is overviewed in Section 1.6.

1.1 Problems and Motivations

Random numbers are used in several fields of computer science and information technology, such as Cryptography, computer games, Monte Carlo Simulation Method, and so on. The random numbers, in cryptography aspect, are the numbers that cannot be predicted. Formally, the sequence of digits is random if and only if the adversaries cannot predict the next digit after seeing all of the previous digits. The unpredictable numbers can be obtained from physical phenomena, such as coin flipping, short noise, and nuclear decay radiation. These truly random numbers require special hardware for extracting from physical phenomena. The physical phenomena is the important factor that restricts the number of truly random outputs from hardware extractor. Therefore, the hardware extractor can produce a limited number of truly random numbers per second. This problem can be seen in applications that use `/dev/random` in Unix-like operating system. These applications are halted by the operating system when there is not sufficient entropy in the entropy pools. In order to increase the throughput, the hardware extractor is used to generate the short truly random sequence. The truly random sequence will be used as a seed for cryptographically secure pseudorandom generator.

Cryptographically secure pseudorandom generator is the algorithm that takes the short uniform random sequence and expands them to the longer sequence that is hard to predict. The expanded se-

quence is called pseudorandom number. The pseudorandom numbers can be predicted, which is different from real random numbers, but it cannot be predicted in a short time. Many pseudorandom generators have been proposed. Some generators generate the pseudorandom number with well statistical properties, but they are not suitable for cryptographic purpose. It is essential to see the difference between the meanings of pseudorandom numbers for normal programming purpose and for cryptographic purpose. In the normal programming purpose, such as simulation application, these pseudorandom numbers need to be reasonably random-looking and have good statistical properties. In contrast, the pseudorandom number for cryptographic purpose must be inefficiently predicted, given a large amount of computational power. The following subsections are stated the problems of using cryptographically secure pseudorandom generators. Then, the motivations to solve these problems are presented in the end of this section.

1.1.1 Performance Problem

Cryptographically secure pseudorandom generators generally require a lot of execution time since they need the complex circuits for performing mathematical functions. For example, RSA generator requires the computational architecture that can perform modular arithmetic over (at least) 1024 bits integers. The lack of performance in cryptographically secure pseudorandom generators is the inevitable problem because they are constructed based on the hardness of some mathematical assumptions. The optimization by decreasing the security parameter's size also affects the overall security of the generators.

1.1.2 Security Problem

The pseudorandom numbers must be inefficiently predicted, given the restricted computational power. However, the processing speed is exponentially increasing that is related to Moore's law. It is necessary to increase the security parameter for enhancing the security level that directly affect to the generator's performance. The quantum computer technology also influences to the security of pseudorandom generators. Moreover, many pseudorandom generators are constructed based on the hardness of factoring assumption. This is false assumption when considering in the quantum algorithm paradigm. The factoring problem can be efficiently solved by Shor's algorithm. Therefore, the pseudorandom generator that remains secure against the attack by a quantum computer is required.

1.1.3 Side-channel Attack on Classical Pseudorandom Generators

Cryptographic applications usually assume that the secret key is confidentially kept and uniformly distributed. This is false assumption since the real-world adversaries have capabilities to manipulate the distribution of the secret keys. They can collect the secret key's information by many methods, such as timing attacks, memory attacks, power analysis attacks, and fault injection attack. This type of attacks is called as the side-channel attack. The most common methods for protecting side-channel attacks is ad hoc and cannot permanently solve these problems. The sustainable method for protecting the side-channel attack is to design the cryptosystem based on the hardness assumption that remains secure in the secret key leakage environment.

1.1.4 Motivations

In order to address these problems, this thesis attempts to construct the hybrid generator between cryptographically secure pseudorandom generator and stream cipher. This is the trade-off between performance and security. This thesis also studies the cryptanalysis on stream cipher and pseudorandom generator, especially the distinguishing attack. Moreover, the cryptographically secure pseudorandom generator which can resist the cryptanalysis by a quantum computer is proposed in this thesis. The cryptographic hardness assumption which can resist the side-channel attacks is studied in this thesis as well.

1.2 Research Objectives

The objectives of this dissertation are as follow.

- To propose an alternative method for performing the cryptanalysis on stream cipher in the machine that is restricted the computational resources.
- To propose a new hybrid stream cipher that reduces the calculation complexity of Blum-Blum-Shub generator and obtains higher security over RC4A.
- To find the cryptographically secure pseudorandom generator which can resist against the distinguishing attack by any adversaries with the excessive computational resources.
- To solve the open question about the security of cryptographic assumption when the secret key is not uniformly selected or the secret key is leaked.

1.3 Scope and Limitations

1. This thesis mainly concerns about the theoretical of distinguishing attack and provable secure pseudorandom generator.
2. The performance simulations in this dissertation do not consider the performance of programming language and compiler that are used to implement the proposed methods.
3. The construction of pseudorandom generator does not consider the leakage of internal stage in each iteration.

1.4 Contributions

Main contributions of this dissertation are as follows.

1. A strong distinguishing attack framework that can precisely calculate the number of samples and the obtained advantage.
2. The hybrid stream cipher that can trade-off between security and performance.
3. The cryptographically secure pseudorandom generator that remains secure against the quantum distinguishing attack.
4. Solving the open question about the robustness of Learning Parity with Noise assumption.

1.5 Methodology

In order to achieve the defined objectives, the following tasks will be stated by mean of appropriate principal related work and theoretical techniques.

1. Study concepts of related mathematical theories and algorithms, such as polynomial-time indistinguishability, pseudorandom generators, stream ciphers, cryptanalysis on stream ciphers, and cryptographic hardness assumptions.
2. Define problems of study. These problems consist of the overestimating/underestimating the number of samples for distinguishing attack, the complexity of cryptographically secure pseudorandom generator, and the security of the pseudorandom generator.

3. Define the architectures of the proposed methods. The proposed architectures attempts to solve the problems of using pseudorandom generator.

4. Developing new algorithms base on the defined architectures.

5. Implementing the proposed methods. The proposed methods are implemented by C, Java, and Python.

6. Testing the keystream's statistical properties and proving the security of the proposed methods. The statistical tests mostly done by National Institute of Standards and Technology test suite. The security proofs mainly based on the reducing argument.

7. Writing the dissertation. The dissertation will conclude overall concepts and techniques which are used in this research.

1.6 Dissertation Organization

This thesis is organized as follows. Chapter II provides the definitions and theories about the pseudorandom generator, and the related works. The alternative approach of strong distinguishing attack is presented in Chapter III. Chapter IV proposes the hybrid algorithm between the stream cipher and the cryptographically secure pseudorandom generator. Chapter V presents the construction of provable secure pseudorandom generator based on the hardness assumption of Learning with Error problem. Chapter VI proves the robustness of Learning Parity with Noises assumption. The overall results and discussions are presents in Chapter VII. The conclusion of this dissertation and the future works are presented in Chapter VIII.

CHAPTER II

BACKGROUND AND RELATED WORKS

The background of this research is presented in Section 2.1. The related works are presented in Section 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, and 2.8.

2.1 Background

Randomness is the essential resource in cryptographic applications. It is used as a keys, passwords, salts, nonces (only once used number), padding byte, blinding value, random challenges for authentication, and so on. The randomness that is used in cryptographic applications can be divided in to two types: the truly random, and the pseudorandom. The truly random numbers can be obtained from non-deterministic sources, such as radio active decay, air turbulence in hardware, and so on. The amount of truly random numbers from non-deterministic sources is insufficient for using in the real world cryptosystems. Although there are sufficient truly random numbers, they may not uniformly distribute and improper to use in cryptographic applications. The randomness extractor usually uses to extract the short uniform sequence from a long truly non-uniform sequence.

To overcome this problem, a pseudorandom generator is used for generating the random numbers instead of using only the truly random number. The pseudorandom generator takes the short uniformly distributed random number and expands them to the long sequence of digits that is hard to distinguish from a real uniform random sequence with the same length. A stream cipher can be categorized as the subclass of the pseudorandom generator. It is a symmetric key cryptosystem where plaintext digits are combined with pseudorandom digits. The stream cipher plays an important role in high speed encryption and wireless applications. However, the pseudorandom generator and the stream cipher cannot produce the real random number. The primitive method for attacking the random numbers from pseudorandom generators and stream ciphers is the distinguishing attack. The objective of distinguishing attack is to find non-randomness in the pseudorandom sequence.

In order to explain the security of the pseudorandom generator, it is worthwhile to formalize the security's definitions. There are two types of security's definitions: *information-theoretic security*, and *computational complexity security*. A cryptosystem is an information-theoretically secure if its security proof is derived from an information theory. The information-theoretic security

mainly considers on the quantity of plaintext's information in the ciphertext. The adversaries simply cannot break the cryptosystem since they have insufficient plaintext's information. Therefore, the information-theoretic cryptosystem is secure even confront with the unbounded computational power adversaries. Informally, the cryptosystem is insecure, in the information-theoretic security aspect, if there exist plaintext's information in the ciphertext. It is possible to construct the cryptosystem that is satisfied by the information-theoretic security definition; however, it is impractical. The interesting information-theoretically secure cryptosystem is One-time pad, which is described in Section 2.2.

The cryptosystem is a computational complexity secure if its security proof is derived from the computational hardness assumption, such as Integer Factorization, RSA problem, Quadratic Residuosity Problem, Discrete Logarithm Problem, and so on. The computational complexity security mainly considers on the infeasibility of extracting the plaintext's information, given the ciphertext and the restricted computational power. Informally, the computational complexity security allows the ciphertext contains the plaintext's information but the plaintext's information must be inefficiently extracted.

This thesis only concerns the pseudorandom generator that is secure under computational complexity security. The cryptographically secure pseudorandom generator must be proven that the breaking cryptosystem problem is at least as hard as solving the computational hardness problem.

2.2 One-time Pad

One-time pad (OTP) is the provable information-theoretically secure symmetric key cryptosystem. Shannon proved OTP's theoretical security in his famous paper in the field of information theory [1] [2]. When properly used, the adversaries, with infinity computational power and infinity time, cannot recover the plaintext after seeing the ciphertext. The ciphertext from OTP is impossible to be cracked because the ciphertext does not contain any plaintext's information. This property is called *perfect secrecy*. The OTP's secret key must be exactly the same size as the plaintext. Moreover, secret key must be real random and can be used only once. Suppose $U = \{U_n\}_{n\in\mathbb{Z}}$ is the uniform distribution over $\mathbb{Z}_2^n$. Assume that $\mathbf{p} \in \mathbb{Z}_2^n$ and $\mathbf{c} \in \mathbb{Z}_2^n$ are plaintext and ciphertext. Let $\mathbf{k} \leftarrow U_n$ be the OTP's secret key. The OTP scheme is defined as $\mathbb{E} = (K, E, D)$ such that:

- **Parameter**. The size of plaintext $n = |\mathbf{p}|$.
- **Secret Key Generation** K. On input 1^n, outputs a uniform random secret key $\mathbf{k} \leftarrow U_n$.

- **Encryption Algorithm** E. On input a secret key $\mathbf{k}$ and a message $\mathbf{p} \in \mathbb{Z}_2^n$,

$$E_{\mathbf{k}}(\mathbf{p}) = \mathbf{p} \oplus \mathbf{k}.$$

- **Decryption Algorithm** D. On input a secret key $\mathbf{k}$ and a ciphertext $\mathbf{c} \in \mathbb{Z}_2^n$,

$$D_{\mathbf{k}}(\mathbf{c}) = \mathbf{c} \oplus \mathbf{k}.$$

Let $\mathcal{P}$, $\mathcal{C}$, and $\mathcal{K}$ be the finite sets of plaintext, ciphertext and secret key, respectively. Assume $p \in \mathcal{P}$ is the plaintext that is sent and $c \in \mathcal{C}$ is the ciphertext that is received. The *priori probability*, $\Pr[p]$, is the probability that the plaintext p is sent. The *posteriori probability*, $\Pr[p|c]$, is the conditional probability of the plaintext p is sent given the ciphertext c is received. The formal definition of perfect secrecy is as follow.

Definition II.1 *(Perfect Secrecy)*
The cryptosystem holds the perfect secrecy property if and only if the priori probability is equal to the posteriori probability. Namely,

$$\Pr[p|c] = \Pr[p] \quad \textit{for every} \quad p \in \mathcal{P} \ \textit{and} \ \ c \in \mathcal{C}.$$

The next theorem proves that OTP has the perfect secrecy property.

Theorem II.1 *One-time pad has the perfect secrecy property.*

proof: To prove this theorem, it is sufficient to show that OTP satisfies the property in Definition II.1. By conditional probability,

$$\Pr[p|c] = \frac{\Pr[p \cap c]}{\Pr[c]}. \tag{2.1}$$

The probability of received ciphertext c is taken overall set of plaintext $\mathcal{P}$ and the probability of the secret key. The OTP's key consists of n independent uniform distributed random bits. Let $k \in \mathcal{K}$ be the secret key that is used. Thus,

$$\Pr[c] = \sum_{p \in P} \Pr[p] \times \Pr[k] = \frac{\sum_{p \in P} \Pr[p]}{2^n} = 2^{-n}. \tag{2.2}$$

Consider the event that the ciphertext c is received, this event depends on the secret key k that is used. Thus,

$$\Pr[p \cap c] = \Pr[p \cap k]$$

The choice of the secret key is independent from the choice of the plaintext. Thus,

$$\Pr[p \cap k] = \Pr[p] \times \Pr[k] = \Pr[p] \times 2^{-n}. \tag{2.3}$$

From Equation 2.1, 2.2, and 2.3, the posteriori probability is

$$\Pr[p|c] = \frac{\Pr[p] \times 2^{-n}}{2^{-n}} = \Pr[p]$$

Thus, OTP is the perfect secure cryptosystem since the priori probability is equal to the posteriori probability. □

However, OTP is generally impractical for the following reasons:

1. High-quality random numbers are difficult to generate.
2. It requires a large amount of memory for storing the secret key.
3. The secret key can be used only once. The reuse of a secret key makes OTP completely break.
4. It is inconvenient to distribute the large secret key.

2.3 Computational Indistinguishability

The computational indistinguishability (or polynomial-time indistinguishability) is the basis of many researches in the field of pseudorandom and cryptography [3] [4] [5]. Its notion is systematically specified on the computational complexity secure definition. Let $\{x_n\}$ and $\{y_n\}$ be the sets of bit sequence length n. Informally, $\{x_n\}$ and $\{y_n\}$ are computationally indistinguishable if there is no probabilistic polynomial time (PPT) algorithm can distinguish them with the non-negligible advantage. The formal definition of computational indistinguishablility and its related definitions are specified as follow.

Definition II.2 *(The Distribution Ensemble)*
Assume I is a (countable) index set. The family of random variables or distributions $X = \{X_i\}_{i \in I}$, X_i is a random variable, is an distribution ensemble indexed by I.

The countable index set I is usually specified to the set of natural number, $\mathbb{N}$. In this dissertation, a distribution ensemble of the form $X = \{X_n\}_{n \in \mathbb{N}}$ contains each X_n ranging over the strings of length $P(n)$, where $P(n)$ is the positive polynomial of n. The uniform ensemble $U = \{U_n\}_{n \in \mathbb{N}}$ is the distribution ensemble, uniformly distributed over strings of length n. Specifically, $\Pr[U_n = x] = 2^{-n}$, for all $x \in \{0,1\}^n$.

Definition II.3 *(Computational Indistinguishability or Polynomial-Time Indistinguishability)*
Assume $n \in \mathbb{N}$ is the security parameter. The distribution ensembles, $\{D_n\}_{n \in \mathbb{N}}$ and $\{E_n\}_{n \in \mathbb{N}}$,

are computationally indistinguishable if for every PPT algorithm D, and every positive polynomial $P(n)$:

$$|\Pr[D(x) = 1 : x \leftarrow D_n] - \Pr[D(y) = 1 : y \leftarrow E_n]| < \frac{1}{P(n)}.$$

The definition of computational indistinguishability by several samples is shown as follow.

Definition II.4 *(Computational Indistinguishability by several samples)*
Assume $n \in \mathbb{N}$ is the security parameter. The distribution ensembles, $\{D_n\}_{n\in\mathbb{N}}$ and $\{E_n\}_{n\in\mathbb{N}}$, are computationally indistinguishable by several samples if for every PPT algorithm D, every positive polynomials $P(n)$ and $Q(n)$, and every indices i such that $1 \leq i \leq Q(n)$:

$$|\Pr[D(x_i) = 1 : x_i \leftarrow D_n] - \Pr[D(y_i) = 1 : y_i \leftarrow E_n]| < \frac{1}{P(n)}.$$

There is the special case of ensembles that are called pseudorandom ensembles. The formal definition of pseudorandom ensembles is shown as follow.

Definition II.5 *(Pseudorandom Ensembles)*
Assume $n \in \mathbb{N}$ is the security parameter. Let $X = \{X_n\}_{n\in\mathbb{N}}$ be an ensemble and let $U = \{U_n\}_{n\in\mathbb{N}}$ be a uniform ensemble. The ensemble X is called pseudorandom if X and U are computationally indistinguishable.

2.4 Pseudorandom Generator

A pseudorandom generator (PRG) is the mathematical function that stretches a real short random sequence to the longer sequence which is hard to distinguish from the uniform distribution. The formal definition of the pseudorandom generator is as follow.

Definition II.6 *(Pseudorandom Generator's standard definition)*
Let $G : \{0,1\}^* \to \{0,1\}^*$ be the PPT algorithm. Then, G is called pseudorandom generator if the following two conditions are satisfied.

1. *Expansion: Let $l : \mathbb{N} \to \mathbb{N}$ be the positive function of n such that $l(n) > n$ for all $n \in \mathbb{N}$. For every $s \in \{0,1\}^*$, $|G(s)| = l(|s|)$.*
2. *Pseudorandomness: The ensemble $\{G(x)\}_{n\in\mathbb{N}}$ is pseudorandom ensemble, where $x \leftarrow U_n$.*

The input x to the pseudorandom generator is called *seed*. The expansion factor of the pseudorandom generator is the differential of $l(n)$ and n. Namely, the expansion factor $\mathrm{Expand}(x) =$

$|G(x)| - |x|$. The pseudorandom requires the input seed is uniformly distributed over the set of string length $n \in \mathbb{N}$. In this thesis, the terms pseudorandom generator and cryptographically secure pseudorandom generator are used in the same meaning.

2.5 Stream Ciphers

A stream cipher is a symmetric-key cryptosystem. It was inspired by Shannon's work on the perfect secrecy and One-time pad. The stream cipher takes two inputs, the secret key and Initialization Vector (IV), and produces the bit sequence called the keystream. The keystream is combined with the plaintext bit by bit, using XOR operation, to produce the ciphertext. Generally, the stream cipher contains less hardware circuit and executes faster than the block cipher. It also has other attractive characteristics over a block cipher, such as limited error propagation [1] and suitable for the devices which have limited memory resource. Stream cipher closely relates to the pseudorandom generator since the stream cipher's security is depended on the wellness of the keystream that imitates the uniformly distributed sequence. Different from the pseudorandom generator, stream ciphers are designed to maintain high performance. It may be vulnerable to distinguishing attack (which will be described in Section 2.6) since it does not rely on the hardness assumption.

Stream cipher is generally modeled into 3 parts:

1. *The Key Scheduling Algorithm (KSA)*. The Key Scheduling Algorithm takes a secret key as the input and produces the stream cipher's internal state. Without loss of generality, the internal state can be modeled as the finite set of bits. The internal state will be used to generate the keystream by the Keystream Generation Algorithm.

2. *The IV Scheduling Algorithm (IVSA)*. The IV Scheduling Algorithm takes the public IV and stream cipher's internal state as the inputs. Then, it updates the internal state, depended on the IV. Note that, some stream ciphers do not have an identical IVSA but they integrate the IVSA with the KSA.

3. *The Keystream Generation Algorithm (KG)*. The Keystream Generation Algorithm takes the internal state from KSA or IVSA as an input and produces the keystream. The KG algorithm can be seen as the iteratively updating algorithm of the internal state. In every iteration, it produces fix-length keystream's bits/bytes, using the internal state, and updates the internal state.

[1] This property only holds in synchronous stream cipher.

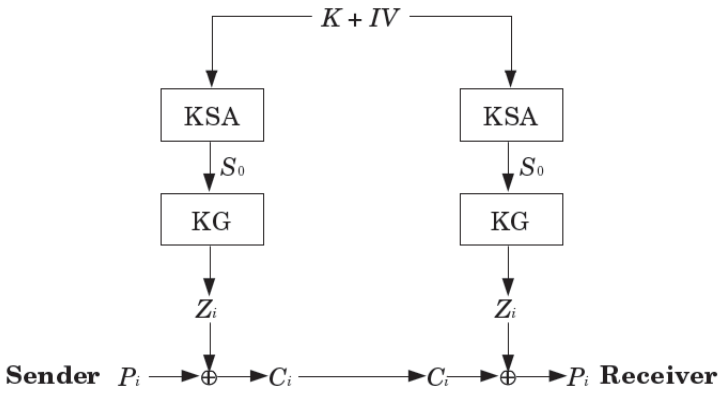


Figure 2.1 Binary Additive Stream Cipher

The general structure of the stream cipher is illustrated in Figure 2.1. The sender and the receiver must share the same key to generate the same keystream since the stream cipher is the symmetric encryption/decryption scheme.

There are two types of the stream cipher: a synchronous stream cipher, and an asynchronous stream cipher. The classification is depended on the method for updating its internal state. In the following subsections, Synchronous and Asynchronous stream ciphers are overviewed.

2.5.1 Synchronous Stream Ciphers

A synchronous stream cipher generates the keystream independent from the plaintext and ciphertext. More specifically, the update internal state algorithm in KG doesn't use the plaintext or ciphertext as an input for updating the internal state. Many stream ciphers are fall into this category, such as alleged RC4, the Py and Py6 Family [6] [7] [8], HC-256 [9], and so on. Suppose that the internal state S_0 is the output of KSA, given the secret key K and the initial vector IV. Let f, g, and h be the state update function, key generation function, and output function, respectively. For each iteration $i = 0, 1, 2, \ldots$, the following equations represent the synchronous stream cipher:

$$k_i = g\left(S_i, K, IV\right)$$
$$c_i = h\left(k_i, p_i\right)$$
$$S_{i+1} = f\left(S_i, K, IV\right).$$

Note that, the additive binary stream cipher normally defines the output function as $h\left(k_i, p_i\right) = k_i \oplus p_i$.

The synchronous stream cipher has smaller error propagation since the error bit cannot affect to the decryption process. However, the deletion and insertion some bits to the ciphertext of syn-

chronous stream cipher make the decryption process fails. The sender and receiver need the perfect synchronization to solve this problem. The another method to overcome this problem is to split the plaintext into frames. Each frames is encrypted by the same key but different IV.

2.5.2 Asynchronous Stream Ciphers

An asynchronous stream cipher takes the previous ciphertexts as an input of KG. Therefore, the keystream from the asynchronous stream cipher depends on the previous ciphertext. The stream cipher that falls into this category is HELIX [10]. It can be represented in the following equations:

$$k_i = g\left(S_i, K, IV\right)$$
$$c_i = h\left(k_i, p_i\right)$$
$$S_{i+1} = f\left(S_i, K, IV, c_i\right)$$

Obviously, the asynchronous stream cipher suffers from the error propagation problem since only one bit flip affects to the receiver's ability to decrypt the ciphertext. The subclass of the asynchronous stream cipher, *self-synchronous stream cipher*, is designed to overcome this problem. The keystream from the self-synchronous stream cipher depends on the limited number of ciphertexts. The stream ciphers that fall into this category are MOSQUITO [11], MOUSTIQUE [12], and SSS [13].

The asynchronous stream cipher has an advantage over the synchronous stream cipher in case of the synchronization loss. The large error propagation makes the asynchronous stream cipher is easier to detect any modifications on the ciphertext. The insertion and deletion some bits to the ciphertext of the self-synchronous stream cipher are harder to detect than the case of the synchronous stream cipher. The design of the secure self-synchronous stream cipher still opens.

2.6 Distinguishing Attacks

A distinguishing attack can be classified as the known plaintext attack. This attack attempts to tell apart the stream cipher's keystream from the real random bit stream. The term *distinguisher* refers to algorithm that performs the distinguishing attack. The easiest approach is to test the keystream by various statistical tests. There are many statistical test suites, such as National Institute of Standards and Technology (NIST) statistical test suite [14], DIEHARD, DIEHARDER, and so on. This approach is not powerful since these test suites are designed for testing general statistical properties and do not aim to the specific stream cipher. More powerful distinguishing attack can be obtained by analyzing the target stream cipher's architecture. This dissertation defines the distinguisher that

takes only one input keystream as a *weak distinguisher* and the distinguisher that takes several input keystreams as a *strong distinguisher*.

The general model of the distinguishing attack is presented in the following subsection. Then, some important distinguishing attacks are reviewed. Pearson's Chi-square test also reviewed in this section since many distinguishing attack mainly rely on testing the samples using Pearson's Chi-square test.

2.6.1 Distinguishing Attack Model

This model represents the general concept of a strong distinguisher. The model of weak distinguisher is omitted since it can be seen as the special case of the strong distinguisher model when the number of samples is 1. The input keystreams to the strong distinguisher are represented by $M \times N$ matrix, where M is the number of keystreams and N is the number of bits in each keystream, such that:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_M \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1N} \\ \vdots & \ddots & \vdots \\ x_{1M} & \cdots & x_{MN} \end{bmatrix}$$

where $x_{ij} \in \{0, 1\}$ for all $1 \leq i \leq M$ and $1 \leq j \leq N$.

Suppose D is a strong distinguisher that takes the matrix $\mathbf{X}$ as an input. Let $\mathcal{G}$ denote the distribution of stream cipher and $\mathcal{U}$ denote the uniform distribution. The distinguisher D has an advantage $\mathrm{Adv}\,(M, N)$ on distinguishing the input $\mathbf{X}$ if and only if

$$\left|\Pr\left[D\left(\mathbf{X}\right) = 1 : \mathbf{X} \leftarrow \mathcal{G}^{M\times N}\right] - \Pr\left[D\left(\mathbf{Y}\right) = 1 : \mathbf{Y} \leftarrow \mathcal{U}^{M\times N}\right]\right| > \mathrm{Adv}\,(M, N)$$

The output bits from a stream cipher usually contain some biases. These biases are represented in the individual random variable x_{ij} in matrix $\mathbf{X}$. It is sufficient to construct the function for transforming the input matrix $\mathbf{X}$ to some useful random variables that represent the biases in keystream samples. These bias random variables will be used to decide that the inputs are sampling from the stream cipher distribution or the uniform distribution. Therefore, the challenge for the distinguishing attack is to find the efficient transfrom function of keystream samples.

2.6.2 Practical Situation

Assume that Alice and Bob use the stream cipher to protect their communications. Let the message sent between Alice and Bob be one of the messages in set $P = \{p_1, p_2, ..., p_t\}$. The ciphertext, sending over the insecure channel, is encrypted in form of $c_i = p_i \oplus k$, for some $1 \leq i \leq t$, and

k is the keystream which is generated from the stream cipher. Assume that Eve, the adversary, has an ability to capture the communication messages between Alice and Bob and she knows the set of all possible messages, P. Eve's task is to decide that the captured ciphertext, c, is encrypted from which one of the plaintext in P. Eve starts the attack scenario by keep XORing each plaintext with captured ciphertext and appling to the distinguisher. Let p_j be the plaintext that Eve chooses from P. Assume that $p_i \oplus p_j$, for all $i \neq j$, is uniform distributed. There are two possible outcome as follow:

- If $i = j$, then $\hat{k} = p_i \oplus k \oplus p_i = k$. Consequently, $\hat{k}$ is distributed according to the stream cipher's distribution which is not uniform.
- If $i \neq j$, then $\hat{k} = p_i \oplus k \oplus p_j$. Consequently, $\hat{k}$ is uniform distributed.

If the distinguisher decides that $\hat{k}$ is not distributed according to the uniform distribution, then Eve will know that the sended plaintext is p_j.

2.6.3 Linear Distinguishing Attacks

The linear distinguishing attack is based on a linear cryptanalysis technique for a block cipher [15]. It was adapted to attack the stream cipher by Golic in [16]. It successfully attacks many stream ciphers, such as Bluetooth stream cipher [17], SOBER-128 [18], SNOW [19] [20], and so on. This attack constructs the samples by applying the keystream bits to the linear boolean function. First, the attacker examines the nonlinear part of the stream cipher and replacing by the linear approximation function. The nonlinear part can be modeled as the linear function and adding with some noise. Then, the attacker finds the linear relationship between the keystream bits. The samples are constructed from the linear function of keystream adding with noises. Normally, the noise's distribution is non-uniform which also causes the samples are non-uniform. These non-uniform samples are used to distinguish the keystream. The linear distinguishing attack can be applied to the class of the stream cipher that based on combining the set of Linear Feedback Shift Registers (LFSR) with the nonlinear boolean function.

2.6.4 Distinguishing Attacks for Array-Based Stream Ciphers

The LFSR-based stream ciphers are appropriated for hardware implementation. In contrast, the array-based stream ciphers are designed for efficient implementing in software. The most well known array-based stream cipher is RC4. RC4 inspires many modern array-based stream ciphers, such as GGHN [21], VMPC [22], RC4A [23], Py-Family stream cipher, Scream [24], MUGI [25], HC-128

and HC-256, and so on. The distinguishing attack framework for the array-based stream cipher is proposed in [26]. This attack starts by examining the internal state which can cause some non-uniform outputs. Let E_i and E_o be an internal state's event and an output's event. The attacker searches the event E_i that affects the occurrence of event E_o, such that $Pr[E_o|E_i] = 0$ or $Pr[E_o|E_i] = 1$. Assume that $Pr[E_o|\neg E_i] = Pr[E_o]$, the probability of event E_i is

$$Pr[E_o] = Pr[E_o|E_i]Pr[E_i] + Pr[E_o|\neg E_i]Pr[\neg E_i] \tag{2.4}$$

The occurrence of event E_o from Equation 2.4 is non-uniform because it is affected by the event E_i. If the $Pr[E_o|E_i]$ close to 1, the event E_o will appear more often. In contrast, the event E_o appears less often when $Pr[E_o|E_i]$ is close to 0.

2.6.5 Chosen IV Distinguishing Attack

A framework for chosen IV distinguishing atttack is proposed in [27]. This framework generalizes the technique calls *d*-monomial test proposed in [28]. The *d*-monomial test considers the keystream as a Boolean function of IV for a fixed key. Assume $IV = iv_1iv_2...iv_m$ is the public IV value, z is the output bit, and f is the boolean function in Algebraic Normal Form (ANF) of IV. The attack starts by consturcting the boolean function $z = f(iv_1, iv_2..., iv_n)$ using the truth table. Then, the attacker counts the number of monomials of weight d in f and compares to the expected value using χ^2-Goodness of fit test with one degree of freedom. However, the *d*-monomail test cannot captures the statistic deviations when the parameter d is close to 0, or the IV's size. The generalized approach in [27] uses P different boolean functions. This approach individually checks the occurence of each monomail and applies to χ^2-Goodness of fit test with 2^n degree of freedom.

2.6.6 Pearson's Chi-square Test

The strong distinguisher usually tests the samples by Pearson's chi-square test. It tests the hypothesis that the input samples are drawn from some known distribution, denoted as $\mathcal{D}$. Let $z_1, z_2, ..., z_r$ be the independently and identically distributed (i.i.d.) random variables, $O(z_i)$ denote the number of outcome of z_i, and $E(\mathcal{D})$ denote the expected number of outcome according to distribution $\mathcal{D}$. The distribution

$$\chi^2 = \sum_{i=1}^{r} \frac{(O(z_i) - E(\mathcal{D}))^2}{E(\mathcal{D})}$$

convergences to the chi-square distribution with degree of freedom r, χ^2_r. The two hypotheses are

H_0 : The random variables are drawn from $\mathcal{D}$.

H_1 : The random variables are not drawn from $\mathcal{D}$.

Let α be the significant level of the test, the hypothesis is rejected if calculated χ^2 is greater than the tabulated $\chi^2(1-\alpha, r)$.

2.7 Learning with Errors

Learning with Errors problem (LWE) was firstly proposed by Regev [29]. This section introduces the basic definition of LWE, the hardness of LWE, and the algorithms for solving LWE. Then, the cryptographic applications from LWE are reviewed at the end of this section.

Definition II.7 *(Learning with Errors)*
Let $n \in \mathbb{Z}^+$ be the security parameter, $m \in \mathbb{Z}^+$ be the number of equations, $q \in \mathbb{Z}^+$ be the modulus, $\mathbf{x} \xleftarrow{\$} \mathbb{Z}_q^n$ be the secret vector, and $\mathcal{X}$ be the noise distribution over $\mathbb{Z}_q$.

- *Decisional LWE Problem: For every distinguisher D with time complexity t, the decisional version of LWE is (n, t, ε) hard if and only if,*

$$|\Pr[D(\mathbf{A}, (\mathbf{A}\mathbf{x} + \mathbf{e}) \bmod q) = 1] - \Pr[D(\mathbf{A}, \mathbf{y}) = 1]| \leq \varepsilon$$

- *Search LWE Problem: For every adversary algorithm A with time complexity t, the search version of LWE is (n, t, ε) hard if and only if,*

$$\Pr[A(\mathbf{A}, (\mathbf{A}\mathbf{x} + \mathbf{e}) \bmod q) = \mathbf{x}] \leq \varepsilon$$

where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$ is the uniform sampling matrix, $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_q^m$ is the uniform sampling vector over $\mathbb{Z}_q^m$, and $\mathbf{e} \leftarrow \mathcal{X}^m$ is the noise vector sampling from distribution $\mathcal{X}^m$.

The important special case of LWE is Regev's LWE, which will be used to construct a pseudorandom generator in Chapter V. The definition of Regev's LWE is as follow.

Definition II.8 *(Regev's LWE)*
Let $n \in \mathbb{Z}^+$ be the security parameter and $P(n)$ be the positive polynomial of n. Regev's LWE uses the modulus q to be prime number such that $q \leq P(n)$. The noise distribution in Regev's LWE is Gaussian Distribution such that $\mathcal{X} = \mathcal{N}\left(0, \alpha^2 q^2 / 2\pi\right)$ is rounded to the nearest integer and modulo q, for any real $\alpha \in (0, 1)$ and $\alpha q > \sqrt{n}$.

2.7.1 The Hardness of LWE

LWE is believed that it is extremely hard and even the quantum algorithm cannot efficiently solve. The hardness of LWE is based on the hardness assumption of lattice problems, the decisional version of shortest vector problem (GapSVP) and shortest independent vector problem (SIVP). The definitions of lattice, GapSVP, and SIVP are defined as follow.

Definition II.9 *(Lattice)*
Given a set of m *linearly independent basis vectors* $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m\}$ *such that each* $\mathbf{v}_i \in \mathbb{R}^n$ *for all* $1 \leq i \leq m$*. The lattice* $\Lambda \subseteq \mathbb{R}^n$ *is defined as*

$$\Lambda(\mathbf{V}) = \left\{ \sum_{i=1}^{m} x_i \mathbf{v}_i | x_i \in \mathbb{Z} \right\}$$

Definition II.10 *Given a set of basis vectors* $\mathbf{V}$*, the lattice* $\Lambda(\mathbf{V})$*, and* $k \in \mathbb{Z}^+$ *be some positive integer. The symbol* $\lambda_k(\Lambda(\mathbf{V}))$ *denotes the smallest radius* r *such that* $\Lambda(\mathbf{V})$ *contains* k *linearly independent vectors of norm at most* r*.*

Definition II.11 *(Shortest Vector Problem, SVP)*
Given a set of basis vectors $\mathbf{V}$ *of dimension* n*, and the lattice* $\Lambda(\mathbf{V})$*. Assume* $\beta(n) > 0$ *is the polynomial aprroximation factor and* $r \in \mathbb{Q}$ *is rational. Then,* $\mathrm{GapSVP}_\beta(\mathbf{V}, r)$ *is defined as follow:*

$$\mathrm{GapSVP}_\beta(\mathbf{V}, r) = \begin{cases} \mathrm{YES} & \textit{if} \quad \lambda_1(\Lambda(\mathbf{V})) \leq r \\ \mathrm{NO} & \textit{if} \quad \lambda_1(\Lambda(\mathbf{V})) > \beta(n) \cdot r \end{cases}$$

Definition II.12 *(Shortest Independent Vector Problem, SIVP)*
Given a set of basis vectors $\mathbf{V}$ *of dimension* n*, and the lattice* $\Lambda(\mathbf{V})$*. Assume* $\beta(n) > 0$ *is the polynomial aprroximation factor and* $k \in \mathbb{Z}^+$ *is a positive integer. Then,* $\mathrm{SIVP}_\beta(\mathbf{V}, k)$ *asks to find the set of linearly indepedent vectors* $\mathbf{U} = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\} \subset \Lambda(\mathbf{V})$ *such that, for ever* $1 \leq i \leq n$*,* $\|\mathbf{u}_i\| \leq \beta(n)\, \lambda_n(\Lambda(\mathbf{V}))$*.*

There is no polynomial time quantum algorithm that can approximate GapSVP and SIVP within the polynomial approximation factor. The hardness of Regev's LWE was proven that it is quantumly as hard as worse case GapSVP and SIVP. The following theorem describes the hardness of Regev's LWE. The proof of this theorem can be found in [29] [30] [31].

Theorem II.2 *If there exist the polynomial time algorithm for solving LWE, then, there exist polynomial time quantum algorithm for solving GapSVP and SIVP with polynomial approximation factor* $\beta(n) = \tilde{O}(n/\alpha)$*.*

2.7.2 Algorithm for solving LWE

The best known algorithm for solving LWE has an exponential time complexity. The easiest method to solve LWE is kept asking for several equations until the unknown vector $\mathbf{x}$ can be estimated. This method requires $2^{O(n \log n)}$ running time and equations. The primitive method to solve LWE is the maximum likelihood estimation. This method requires $O(n)$ samples, and execute time $2^{O(n \log n)}$. More efficient algorithm for solving LWE is Blum-Kalai-Wasserman (BKW) algorithm. It was proposed by Blum et al. in [32]. This algorithm uses the generalized birthday paradox technique and majority vote for solving LWE. It requires $2^{O(n)}$ running time and samples. Note that, BKW algorithm can be used to solve Learning Parity Noises problem, which will be discussed in the next section. The recent algorithm, proposed by Arora and Ge [33], requires $2^{O((\alpha q)^2)}$ running time and equations. This is the first algorithm which can solve LWE in the sub-exponential time when $\alpha q < \sqrt{n}$.

2.7.3 Applications from LWE

LWE is used to construct several cryptographic applications. Many LWE based leakage-resilient cryptosystems are proposed, such as the leakage-resilient symmetric key cryptosystem and the obfuscator for point functions with multi-bit output [34], the leakage-resilient public key cryptosystem [35]. The pseudo-random function from LWE is proposed in [36]. The oblivious transfer protocol from LWE was proposed in [37]. The LWE based collision resistance hash function was proposed in [38]. The identity-based encryption schemes were proposed in [39] [40] [41].

2.8 Learning Parity with Noise

Learning Parity with Noise (LPN) can be seen as the special case of LWE when the modulus q is 2. The formal definition of LPN is as follow.

Definition II.13 *Let $n \in \mathbb{Z}^+$ be the security parameter, $m \in \mathbb{Z}^+$ be the number of equations, $\mathbf{x} \xleftarrow{\$} \mathbb{Z}_2^n$ be the secret vector, and $\mathcal{X}$ be the noise distribution over $\mathbb{Z}_2$.*

- *Decisional LPN Problem: For every distinguisher D with time complexity t, the decisional version of LPN is (n, t, ε) hard if and only if,*

$$|\Pr[D(\mathbf{A}, \mathbf{A} \cdot \mathbf{x} \oplus \mathbf{e}) = 1] - \Pr[D(\mathbf{A}, \mathbf{y}) = 1]| \leq \varepsilon$$

- *Search LWE Problem: For every adversary algorithm A with time complexity t, the search version of LPN is (n, t, ε) hard if and only if,*

$$\Pr\left[A\left(\mathbf{A}, \mathbf{A}\cdot\mathbf{x}\oplus\mathbf{e}\right)=\mathbf{x}\right]\leq\varepsilon$$

where $\mathbf{A}\xleftarrow{\$}\mathbb{Z}_2^{m\times n}$ is the uniform sampling matrix, $\mathbf{y}\xleftarrow{\$}\mathbb{Z}_2^m$ is the uniform sampling vector over $\mathbb{Z}_2^m$, and $\mathbf{e}\leftarrow\mathcal{X}^m$ is the noise vector sampling from distribution $\mathcal{X}^m$. Normally, the error distribution $\mathcal{X}$ is Bernoulli distribution, which is denoted as Ber_η where $0<\eta<1/2$.

2.8.1 Hardness of LPN

LPN seems easier than LWE since the LPN's modulus is restricted. However, it is believed that there is no efficient algorithm for solving search LPN. LPN can be restated as the decoding random linear codes, which is the NP-complete problem [42]. The efficient algorithm for resolving search LPN problem implies the efficient algorithm for solving problem in the coding theory. Moreover, the hardness of decision and search version of LPN is proven to be polynomially equivalent [43]. Nevertheless, the assumption of inequality between P and NP does not imply that the search LPN problem is hard since LPN is the average-case hardness but the NP-complete is the worse-case hardness.

2.8.2 Algorithm for solving LPN

The most efficient algorithms for solving LPN has exponential time complexity [32] [44] [45]. The BKW algorithm [32] is based on the generalized birthday paradox [46]. The running time of BKW algorithm is $2^{O(n/\log n)}$, given the polynomial of n samples. The LPN solving algorithm in [44] can be seen as the variant of the fast correlation attack against stream ciphers. The authors claim that the proposed algorithm is more efficient in most cases but the success probability is still questioned. The improvement versions of BKW algorithm are proposed in [45]. The first algorithm, LF1, uses the Walsh-Hadamard transform to find the best possible samples; thus, the number of queries is less than BKW algorithm. The second algorithm, LF2, uses the heuristic approach and gains more efficient than LF1 in a practical situation.

2.8.3 Applications from LPN

Although LPN is not as versatile as LWE, LPN attracts many cryptographers since it is simple and it can be efficiently implemented. Many cryptographic applications are constructed based on the hardness of LPN. The efficient pseudorandom generator from LPN was proposed in [47] [48].

The private key encryption scheme from LPN was proposed in [49]. Moreover, the secure against chosen plaintext attack (CPA secure) encryption scheme that still secure with exponentially hard-to-invert auxiliary input was proposed in [50]. LPN was used to construct the well known protocols, HB [51] and its adaptive version HB^{+} [52]. These authentication protocols are mostly used for RFID authenticating. The variant of LPN was used to construct the public key encryption scheme in [53]. However, there is no method for constructing the sophisticated cryptographic application, such as collision resistant hash function or public key encryption scheme, from LPN.

CHAPTER III

WEIGHT-ADJUSTING STRONG DISTINGUISHER

In this chapter, the framework for strong distinguishing attack, which can calculate more accurate number of samples, is presented. The precisely strong distinguisher's advantage can be calculated, given the number of samples and the weak distinguisher's advantage. As a result, the proposed method can clearly show the trade-off between the space complexity and the strong distinguisher's advantage.

This chapter is organized into 6 sections. In Section 3.1, some mathematic notations are described. Section 3.2 presents the general weak distinguisher model and the weight-adjusting strong distinguisher. Section 3.3 gives the experimental result, and shows that the conventional method always underestimate the number of samples. The discussion of advantages of this framework and the conclusion are presented in Section 3.4 and Section 3.5, respectively.

3.1 Notations

For any $n \in \mathbb{N}$, the symbol U_n and $U_{l(n)}$ denote the uniform distribution over $\{0,1\}^n$ and $\{0,1\}^{l(n)}$, where $l(n)$ is the positive polynomial of n. The symbol $x \leftarrow X$ denotes the sampling process of the value x from the distribution X. Let $G : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ be the function that takes the uniform input bits length n and output the sequence length $l(n)$ which is computationally indistinguishable from uniform sequence length $l(n)$. In other words, G is a PRG that takes the seed length n and output the pseudorandom sequence length $l(n)$.

3.2 Weight-adjusting Strong Distinguisher

This section starts by modeling the general PPT weak distinguisher. This model is not only flexible, but it also captures weak distinguisher's properties. It can be shown that any weak distinguishers can be transformed to this model. This model will be used to construct the weight-adjusting strong distinguisher. The number of samples which can satisfy the required strong distinguisher's advantage is shown in the end of this section.

3.2.1 Modeling the General PPT Weak Distinguisher

In this chapter, the weak distinguisher, D, must have properties as defined in the Assumption III.1

Assumption III.1 *Suppose that D is the PPT weak distinguisher for G with advantage ε. Let $x \leftarrow U_n$ and $y \leftarrow U_{l(n)}$ be the discrete random variables. If the distributions of $G(x)$ and y are close to each other, then* $\Pr[D(y) = 1] = \frac{1}{2}$ *and* $\Pr[D(G(x)) = 1] = \frac{1}{2} + \varepsilon$.

There are two reasons why the weak distinguisher with such specific properties is mentioned. First, the random bits from stream ciphers are easily distinguished from the uniform random sequence when examining in the bit level. Second, any weak distinguishers can be transformed to D. This means that D can be used as the substitute of any PPT weak distinguishers. The transform process is shown in the following theorem.

Theorem III.1 *Let D'' be the PPT weak distinguisher with properties* $\Pr[D''(y) = 1] = p$ *and* $\Pr[D''(G(x)) = 1] = p + \varepsilon''$*, where $p \neq \frac{1}{2}$. There exist a distinguisher, D, constructed from D'', which satisfy the properties in Assumption III.1.*

proof: This proof is shown by transforming the distinguisher D'' to D. Suppose $Z_l = z_1 z_2 .. z_l$ is the input bit sequence length l to D. The input sequence to D'' is denoted as $Z_{i-1} r_i R_{l-i} = z_1 z_2 ... z_{i-1} r_i r_{i+1} ... r_l$, where $0 \leq i \leq l$ and the first $i-1$ bits are obtained from D's input sequence and the rest $l-i+1$ bits are uniformly chosen. The construction of distinguisher D from D'' is shown in Algorithm III.1

Algorithm III.1 The transformation of D'' to D

Input: The random bit sequence $Z_l = z_1 z_2 .. z_l$.

Output: Distinguishing the input bit sequence from uniform distribution.

Randomly select i.

$Z_{i-1} r_i R_{l-i} = z_1 z_2 \ldots z_{i-1} r_i r_{i+1} \ldots r_l$

return $D''(Z_{i-1} r_i R_{l-i}) \oplus r_i \oplus z_i$

Assume $x \leftarrow U_n$ and $y \leftarrow U_{l(n)}$ are the uniform random bit strings over $\{0,1\}^n$ and $\{0,1\}^{l(n)}$. The probability when the input sequence to D is real random sequence can be calculated as follows:

$$\begin{aligned} \Pr[D(y) = 1] &= \Pr\left[D''(Z_{i-1} r_i R_{l-i}) = 1 \wedge r_i = z_i\right] + \Pr\left[D''(Z_{i-1} r_i R_{l-i}) = 0 \wedge r_i \neq z_i\right] \\ &= \frac{p}{2} + \frac{1-p}{2} = \frac{1}{2} \end{aligned} \tag{3.1}$$

Let $G_l = g_1 g_2 ... g_l$ be the pseudorandom sequence. There must be at least one value of i such that

$$\Pr\left[D''\left(G_{i-1} g_i R_{l-i}\right) = 1\right] - \Pr\left[D''\left(G_{i-1} r_i R_{l-i}\right) = 1\right] \geq \frac{\varepsilon''}{l}$$

WLOG, assume that $\Pr\left[D''\left(G_{i-1} g_i R_{l-i}\right) = 1\right] = p + \frac{\varepsilon''}{l}$, and $\Pr\left[D''\left(G_{i-1} r_i R_{l-i}\right) = 1\right] = p$. Let $z = \Pr\left[g_i = 0\right]$, and let $a_{jk} = \Pr\left[D''\left(G_{i-1} j R_{l-i}\right) = 1 | g_i = k\right]$. Then,

$$\begin{aligned}
\Pr\left[D''\left(G_{i-1} r_i R_{l-i}\right) = 1\right] = & \quad \frac{\Pr\left[D''\left(G_{i-1} 0 R_{l-i}\right) = 1\right]}{2} + \frac{\Pr\left[D''\left(G_{i-1} 1 R_{l-i}\right) = 1\right]}{2} \\
= & \quad \frac{z a_{00} + (1-z) a_{01}}{2} + \frac{z a_{10} + (1-z) a_{11}}{2} \\
\Pr\left[D''\left(G_{i-1} g_i R_{l-i}\right) = 1\right] = & \quad z a_{00} + (1-z) a_{11}
\end{aligned}$$

Subtracting the previous equations, the advantage ε''/l is

$$\frac{\varepsilon''}{l} = \frac{z(a_{00} - a_{10}) + (1-z)(a_{11} - a_{01})}{2}$$

The probability when D answers 1 for fixing $r_i = 0$ and $r_i = 1$ are

$$\begin{aligned}
\Pr[D''\left(G_{i-1} r_i R_{l-i}\right) \oplus g_i = 1 | r_i = 0] &= \Pr[g_i = 0] \Pr[D''\left(G_{i-1} 0 R_{l-i}\right) = 1 | g_i = 0] + \\
&\quad \Pr[g_i = 1] \Pr[D''\left(G_{i-1} 0 R_{l-i}\right) = 0 | g_i = 1] \\
&= z a_{00} + (1-z)(1 - a_{01}) \\
\Pr[D''\left(G_{i-1} r_i R_{l-i}\right) \oplus g_i = 0 | r_i = 1] &= \Pr[g_i = 0] \Pr[D''\left(G_{i-1} 1 R_{l-i}\right) = 0 | g_i = 0] + \\
&\quad \Pr[g_i = 1] \Pr[D''\left(G_{i-1} 1 R_{l-i}\right) = 1 | g_i = 1] \\
&= z(1 - a_{10}) + (1-z) a_{11}
\end{aligned}$$

The probability when the input to D is the pseudorandom sequence can be calculated as follows:

$$\begin{aligned}
\Pr[D\left(G\left(x\right)\right) = 1] &= \frac{\Pr[D''\left(G_{i-1} r_i R_{l-i}\right) \oplus g_i = 1 | r_i = 0]}{2} + \\
&\quad \frac{\Pr[D''\left(G_{i-1} r_i R_{l-i}\right) \oplus g_i = 0 | r_i = 1]}{2} \\
&= \frac{1}{2}(z a_{00} + (1-z)(1 - a_{01}) + z(1 - a_{10}) + (1-z) a_{11}) \\
&= \frac{1}{2} + \frac{z(a_{00} - a_{10}) + (1-z)(a_{11} - a_{01})}{2} \\
&= \frac{1}{2} + \frac{\varepsilon''}{l} \qquad (3.2)
\end{aligned}$$

From Equation (3.1) and (3.2) , D is the weak distinguisher that satisfies the properties in Assumption III.1 and $\varepsilon = \varepsilon''/l$. $\square$

3.2.2 Construction

The proposed framework uses Binomial theorem for testing the input bit sequence samples instead of χ^2-Goodness of fit test. The proposed strong distinguisher tests the following hypotheses.

H_0 : The samples are uniform distribution.

H_1 : The samples are non-uniform distribution.

The algorithm starts by applying each sample to the weak distinguisher, D, and counts the number of success distinguishing. The total number of success distinguishing will be used in the deciding step of the strong distinguisher. The deciding step examines the difference between the expected value and the number of success distinguishing. The weight adjusting strong distinguisher is presented in Algorithm III.2

Algorithm III.2 Weight-adjusting Strong Distinguisher

Input: The random bit sequence samples $[\alpha_1, \alpha_2, \ldots, \alpha_N]$.

Output: Distinguishing the input bit sequence from uniform distribution.

$c = 0$ $\{c$ is the counter$\}$

for each α_i **in** $[\alpha_1, \alpha_2, \ldots, \alpha_N]$ **do**

  if $D(\alpha_i) = 1$ **then**

    $c = c + 1$

  end if

end for

if $\mid N/2 - c \mid > \sqrt{N}/2$ **then**

  return 1

else

  return 0

end if

3.2.3 Calculating the Strong Distinguisher's Advantage

The data complexity of a strong distinguisher is usually described in form of $N = 2^k$. Because of this reason, this research only considers the even number of samples. The advantage of the proposed strong distinguisher can be defined as a function of number of samples and the weak distinguisher's advantage.

Theorem III.2 *Suppose N is an even number of samples, $c^- = \left\lfloor \frac{N-\sqrt{N}}{2} \right\rfloor$, $c^+ = \left\lfloor \frac{N+\sqrt{N}}{2} \right\rfloor$, and ε is the weak distinguisher's advantage. The advantage function of the weight adjusting strong distinguisher can be defined as* $\mathrm{Adv}\,(N,\varepsilon) = \left| 2^{-N} \sum_{i=c^-}^{c+} \binom{N}{i} \left[1 - (1-4\varepsilon)^{N/2} \right] \right|$.

proof: The strong distinguisher will accept the input sequences when $|N/2 - c| > \frac{\sqrt{N}}{2}$. It means that the strong distinguisher answers 1 when $c < c^- = \frac{N-\sqrt{N}}{2}$ and $c > c^+ = \frac{N+\sqrt{N}}{2}$. Let $F(a,b;N,p) = \sum_{i=a}^{b} \binom{n}{i} (p)^i (1-p)^{N-i}$ be the cumulative binomial distribution from a to b. Assume D' is the proposed strong distinguisher. The false positive probability and the true positive probability of the strong distinguisher are calculated as follow.

$$\begin{aligned}
\Pr[D'(y_1,...,y_N)=1] &= 1 - F(c^-,c^+;N,\frac{1}{2}) = 1 - \sum_{i=c^-}^{c^+} \binom{N}{i} \left(\frac{1}{2}\right)^i \left(\frac{1}{2}\right)^{N-i} \\
&= 1 - \frac{1}{2^N} \sum_{i=c^-}^{c^+} \binom{N}{i}
\end{aligned}$$

$$\begin{aligned}
\Pr[D'(G(x_1),...,G(x_N))=1] &= 1 - F(c^-,c^+;N,\frac{1}{2}+\varepsilon) \\
&= 1 - \sum_{i=c^-}^{c^+} \binom{N}{i} \left(\frac{1}{2}+\varepsilon\right)^i \left(\frac{1}{2}-\varepsilon\right)^{N-i} \\
&= 1 - (\frac{1}{4}-\varepsilon^2)^{N/2} \sum_{i=c^-}^{c+} \binom{N}{i} \left(\frac{\frac{1}{2}-\varepsilon}{\frac{1}{2}+\varepsilon}\right)^{N/2-i} \\
&\approx 1 - \frac{1}{2^N}(1-4\varepsilon^2)^{N/2} \sum_{i=c^-}^{c+} \binom{N}{i}
\end{aligned}$$

The advantage of weight adjusting strong distinguisher is

$$\begin{aligned}
\mathrm{Adv}(N,\varepsilon) &= |\Pr[D'(G(x_1),...,G(x_N))=1] - \Pr[D'(y_1,...,y_N)=1]| \\
&= \left| \frac{1}{2^N} \sum_{i=c^-}^{c^+} \binom{N}{i} \left[1 - \left(1-4\varepsilon^2\right)^{N/2} \right] \right| \qquad (3.3)
\end{aligned}$$

□

3.2.4 Calculating the Number of Samples

Assume that N is the number of samples, ε is the weak distinguisher's advantage, and ε' is the required strong distinguisher's advantage. Let $f : \mathbb{Z}^+ \to \mathbb{R}$ be the function such that $f\,(N) = \mathrm{Adv}(N,\varepsilon) - \varepsilon'$. The number of samples which can satisfy the required advantage can be calculated by solving the equation $f\,(N) = 0$. The algorithm for solving this equation is shown in Algorithm III.3. This algorithm starts by searching the upper bound and lower bound of the number of samples. Then, Bisection method is used for optimizing the number of samples.

Algorithm III.3 Algorithm for Calculating the Number of Samples

Input: MAX is the maximum iterations. The function $f(N) = \mathrm{Adv}(N, \varepsilon) - \varepsilon'$.

Output: The number of samples for strong distinguishing attack.

$N_l = 1, N_u = 4$

while $f(N_l) \times f(N_u) > 0$ **do**

$N_l = N_u, N_u = N_u \times 2$

end while

for $i = 1 \rightarrow MAX$ **do**

$N_c = \lfloor (N_l + N_u)/2 \rceil$

if $f(N_c) \times f(N_l) > 0$ **then**

$N_a = N_c$

else

$N_b = N_c$

end if

$i = i + 1$

end for

return N_c

3.3 Experimental Result

Traditionally, the number of samples is usually estimated to $N \approx \frac{1}{\varepsilon^2}$, when $\Pr[D(G(x)) = 1] = 0.5(1 + \varepsilon)$. This estimated number of samples is applied to the Equation (3.3) and the result shows that the obtained advantage is less than 0.4. In contrast, the weight adjusting distinguisher can adjust the number of samples to satisfy the required advantage using Algorithm III.3. The relation between the advantage and the number of samples, when $\varepsilon = 2^{-8}$ to 2^{-12}, is shown in Figure 3.1. This graph shows that the strong distinguisher's advantage can be precisely computed given the number of samples. Moreover, the graph shows that when the weak distinguisher's bias decreases 2 times, the number of samples have to increase 4 times for preserving the same advantage level.

3.4 Discussions

The strong distinguisher is typically constructed on Pearson's chi-square hypothesis testing. However, it is hard to calculate the actual advantage level, given the number of samples. The number

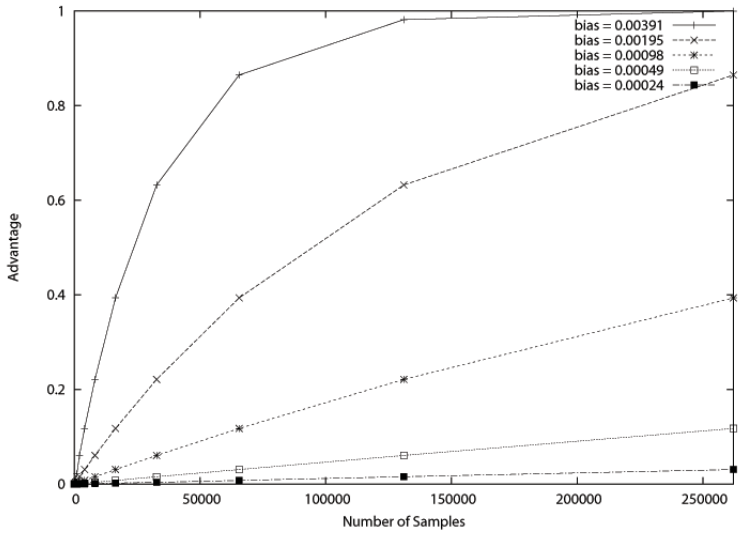


Figure 3.1 The Relation between Strong Distinguisher's Advantage and the Number of Samples.

of samples can be overestimated or underestimated, and not match to the computational resource. In contrast, this research proposes the weight adjusting strong distinguisher framework. The proposed method is based on the binomial hypothesis testing. This approach has the advantage when the computational resource is limited. The number of samples can be determined beforehand to satisfy the resources and the required advantage level. The Equation (3.3) shows the trade-off between the number of samples and the achieved advantage. The space and time complexity are reduced since the number of samples is not overestimated. This model can show the distinguisher's advantage which can be obtained by inadequatewhen number of samples. Moreover, Pearson's chi-square test requires a large amount of samples so that the approximation can be valid. The binomial hypothesis testing is suitable when the number of samples is limited.

3.5 Conclusions

The conventional method for approximating the number of samples always underestimate the number of samples which is used in strong distinguishing attack. Furthermore, there is no method for verifying the achieved advantage, given the number of samples. In this research, the alternative strong distinguisher approach is presented. This adaptation is suitable for the limited resource machine. It can accurately calculate the number of samples that is the trade-off between the advantage and the space complexity. The weight-adjusting strong distinguisher is constructed based on general PPT weak distinguisher model and binomial hypothesis testing. The general PPT weak distinguisher model is proved in Theorem III.1 that it is general enough for capturing any PPT weak distinguisher's

properties. Binomial hypothesis testing is used in the distinguishing step. Consequently, the proposed method is also suitable when the memory is limited since the binomial hypothesis is much accurate than Pearson’s chi-square test under the small sample size.

CHAPTER IV

ENHANCING SECURITY OF STREAM CIPHER BY CRYPTOGRAPHIC SECURE PSEUDORANDOM GENERATOR

This chapter presents the hybrid algorithm between a Cryptographically Secure Pseudorandom Generator(CSPRG) and a stream cipher, the RC4B. The RC4B is the combination of the shuffle arrays based stream cipher, RC4A, and the CSPRG, Blum-Blum-Shub generator (BBS). The proposed method can trade-off between the execution time and the security level. The objectives of the RC4B are to avoid such distinguishing attack for general stream cipher and improving the security of the stream cipher. Moreover, this chapter also shows that the RC4B 's execution time is less than the based CSPRG.

This chapter is organized into 5 sections. The Section 4.1 explains the related stream cipher and CSPRG, RC4A and Blum-Blum-Shub generator. In Section 4.2, the RC4B's algorithm is proposed. The RC4B's performance and security are proposed in the Section 4.3. The discussion of advantages of the RC4B over RC4A and Blum-Blum-Shub generator is presented in Section 4.4. Finally, the conclusion is presented in Section 4.5.

4.1 Preliminaries

This section presents the stream cipher and the cryptographically secure pseudorandom generator that are used as the prototypes of the proposed method, RC4A and Blum-Blum-Shub generator.

4.1.1 RC4A Stream Cipher

RC4A is the variation of RC4 algorithm published by Paul and Preneel [23]. They presented RC4's weakness and proposed RC4A which can resist to their attack. Its byte stream is produced based on the shuffle arrays algorithm. The array's size, N, is selected in form of $N = 2^k$ where $k \in \mathbb{Z}^+$. Practically, the value of k is normally chosen to 8, so the array's size is $N = 2^8$. The main difference between RC4 and RC4A is the number of S-boxes using in the pseudorandom bit generation process. RC4A uses two S-boxes in parallel while RC4 uses only one S-box. Thus, the output bit streams of RC4A will be depended on various random variables. As a result, the correlation

between output bits and hidden states is decreased. However, they have not proposed the new Key-Scheduling Algorithm (KSA) for RC4A and assumed that the initial permutation can work perfectly. RC4A's algorithm is shown in the Algorithm IV.1. Note that, all index calculations are assumed to modulo by the array's size.

Algorithm IV.1 RC4A's Keystream Generator

Require: N be the security parameter such that $N = 2^k$, where $k \in \mathbb{N}$.

$S_1 = [0, 1, \ldots, N-1]$ and $S_2 = [0, 1, \ldots, N-1]$.

$i = j_1 = j_2 = 0$

permute(S_1), **permute**(S_2)

repeat

$i = i + 1$

$j_1 = j_1 + S_1[\, i \,]$

swap$(S_1[\, i \,], S_1[\, j_1])$

output $S_2[\, S_1[\, i \,] + S_1[\, j_1] \,]$

$j_2 = j_2 + S_2[\, i \,]$

swap$(S_2[\, i \,], S_2[\, j_2])$

output $S_1[\, S_2[\, i \,] + S_2[\, j_2] \,]$

until Reach the required amount of random bits

There are two papers report the biases found in RC4A. Both biases concern with the equivalent output bytes. The causes of these biases come from 2 factors: the flaw in RC4A's shuffle algorithm, and the non-uniformity of internal states. The summarizations of both biases are presented as follow.

The bias in the outputs at time t and time $t + 2$, when the known value i is even, was proposed by Maximov [54]. The number of samples used in the distinguisher is 2^{58} samples. This research concludes that although the internal states of RC4 family are uniformly distributed, it is still insecure. The solutions for enhancing the security of RC4 family are to increase the number of shuffle instructions in each generation loop, or abandon some outputs. Nevertheless, these adaptations can directly influence to the PRBG's efficiency.

The serious weakness of RC4A was published in paper [55]. This research found that the probability of the first and the third output bytes are equal is $2^{-8} \times \left(1 - 2^{-8.01}\right)$. They also proved that the amount of data required to distinguish the RC4A's output is $O\left(2^{24.02}\right)$ samples, which is less than the sample required in the previous research. However, the simulation shows that the probability of the first and the third output bytes are equal is $2^{-8} \times \left(1 - 2^{-7.55}\right)$. For this reason, the amount of

data needed for distinguishing is $O\left(2^{23.1}\right)$ because of the flaw in the initial permutation process.

4.1.2 Blum-Blum-Shub Generator

The Blum-Blum-Shub (BBS) or $x^2 \mod n$ generator is the cryptographically secure pseudo-random generator. BBS was constructed based on the hardness assumption of the Quadratic residuosity problem. The problem of distinguishing the BBS's random bits was proved that it is not easier than solving the Quadratic residuosity problem. The security proof of BBS can be found in [56–58]. Some randomness properties of the BBS's sequences were proved in [59]. The BBS's algorithm is shown in the Algorithm IV.2.

Algorithm IV.2 Blum-Blum-Shub's Algorithm

Input: Blum Primes p and q.

Output: The pseudorandom bits.

$n = p \times q$

Uniformly choose s from $\mathbb{Z}_n^*$

$x = s^2 \bmod n$

repeat

  $x = x^2 \bmod n$

  output parity(x)

until Reach the required amount of random bits

4.2 Proposed Method : the RC4B

In this section, the hybrid method among arrays based stream cipher, RC4A, and CSPRG, Blum-Blum-Shub generator, is proposed.

4.2.1 RC4B Design Principles

The outputs from the arrays based stream cipher always leak their internal states information. Due to this weakness, the random streams from the arrays based stream cipher can always be distinguished as shown in [26]. The main idea of the hybrid method is to conceal the RC4A's random streams by the intractable mathematics problem. Therefore, the secret information is masked out and the output stream will be depended on more random variables. This improvement will increase the complexity of distinguishing attack.

4.2.2 RC4B Description

The RC4B's algorithm is the combination of BBS and RC4A. The pseudorandom bit generation method is based on BBS but it extends a permutation on the Blum primes by the RC4A's permutation method. The RC4B uses many small Blum primes to produce many Blum integers instead of using one large Blum integer. Therefore, the BBS's computation complexity is decreased and the RC4A's internal state information is masked out. For convenient in the explanation, the auxiliary abstract data type is defined as follow. Let A be the random variable that containing 2 integers, an $index$ and a $prime$. The $index$ element stores a pointer that points to another address in the array, and the $prime$ element stores a Blum prime. This ADT can use the two operations:

- A.getIndex(): return the A's index value.
- A.getPrime(): return the A's Blum prime.

Each random variable is stored in the arrays P and Q with length N. These two arrays must be permuted based on the key by KSA. Then, they will be used in the RC4B's random bit generation method. However, this work omits the design of KSA for the RC4B due to it beyonds the scope of this research and assumes that the arrays P and Q are uniformly permuted. The explanation of the generation method is described as follows.

The RC4B's keystream generator function requires the permuted arrays, P and Q, and integer seed, s, as the parameters. Then, the permutation process is applied on the arrays and fetches two Blum primes in each round. These Blum primes and seed are used to create a co-prime by the **GenCoprime** method, the variant of unit generation algorithm proposed in [60]. For generating the pseudorandom bits, the operation $x_i = x_{i-1}^2 \mod n$ from BBS is applied to the co-prime. Then, the hardcore bit is used as an output bit. The RC4B's keystream generation algorithm and **GenCoprime**'s algorithm are shown in Algorithm IV.3 and Algorithm IV.4. All index calculations are assumed to modulo by the array's size.

Note that, the Carmichael function, $\lambda(n)$, can be calculated in the polynomial time if the prime factors of n are known. Similarly, the **GenCoprime**'s algorithm can be executed in the polynomial time as well. The performance and security analysis of the RC4B will be presented in the next section.

4.3 Performance and Security Analysis

In this section, the RC4B's computation complexity is calculated, in term of asymptotic notation, and compared to BBS. The experimental result shows that the RC4B requires less execution

Algorithm IV.3 The RC4B's Keystream Generation Algorithm

Require: The arrays P and Q, each has size N and the integer seed, s.

$i = 0, j_1 = 0, j_2 = 0$

repeat

$i = i + 1$

$j_1 = j_1 + P[\, i \,]$.getIndex ()

swap$(P[i], P[j_1])$

$q = Q[P[\, i \,]$.getIndex () $+ P[j_1]$.getIndex ()].getPrime ()

$j_2 = j_2 + Q[\, i \,]$.getIndex ()

swap$(Q[i], Q[j_2])$

$p = P[Q[\, i \,]$.getIndex () $+ Q[j_1]$.getIndex ()].getPrime ()

$n = p \times q$

$s =$ **GenCoprime** (s, n)

$s = s^2 \bmod n$

$x = s^2 \bmod n$

while $x \neq s$ **do**

output parity(x)

$x = x^2 \bmod n$

end while

until Reach the required amount of random bits

Algorithm IV.4 GenCoprime's Algorithm

Require: The integer seed, s, and Blum integer, n.

$s = s \bmod n, c = \lambda\,(n)$

loop

$U = (1 - s^c) \bmod n$

if $U \neq 0$ **then**

Randomly chosen $r \in \mathbb{Z}_n$.

$s = (s + r \times U) \bmod n$

else

return s

end if

end loop

time than BBS. Moreover, the RC4B's security is also analyzed. The result shows that the RC4B gains higher security than RC4A.

4.3.1 Performance Analysis

The RC4B's computation complexity is calculated in term of Big-oh notation and compared to BBS. The calculation is shown in the following theorem.

Theorem IV.1 *Suppose n is the Blum integer which is used in BBS and n' is the largest Blum integer which is used in the RC4B. Let m be the number of random bits produced in one round of the RC4B. The RC4B's time complexity is less than BBS's time complexity if $\|n'\| < \|n\| - \frac{1}{m}$.*

proof: Let $T_{BBS}(x,n)$ is the BBS's time complexity for generating x bits with Blum Integer n. Let $T_{RC4B}(x,n')$ be the RC4B's time complexity for calculating x bits and the largest Blum integer which is used in the RC4B is n'. Obviously, $T_{BBS}(x,n) = xO\left(\|n\|^2\right)$. Let m be the number of random bits produced in one round of the RC4B. The RC4B's time complexity can be calculated as follow.

$$T_{RC4B}\left(x,n'\right) = \sum_{i=1}^{\frac{x}{m}}\left(O\left(1\right) + mO\left(\|n'\|^2\right)\right) = \sum_{i=1}^{\frac{x}{m}} O\left(1\right) + \sum_{i=1}^{\frac{x}{m}} mO\left(\|n'\|^2\right) = \frac{x}{m} + xO\left(\|n'\|^2\right)$$

Let the equation $\|n'\| < \|n\| - \frac{1}{m}$ hold. As a result,

$$\begin{aligned} O\left(\|n'\|^2\right) &< O\left(\|n\|^2\right) - \frac{1}{m} \\ \frac{x}{m} + xO\left(\|n'\|^2\right) &< xO\left(\|n\|^2\right) \\ T_{RC4B}\left(x,n'\right) &< T_{BBS}\left(x,n\right) \end{aligned}$$

Normally, the Blum integers which are used in the RC4B always have smaller size than the Blum integer that is required by BBS. Consequently, the RC4B's time complexity is less than the BBS's time complexity. □

To support the proof of the Theorem IV.1, the execution times of the RC4B and BBS are compared. Both algorithms of the RC4B and BBS are implemented in Java. The BigInteger class is using to handle the arithmetic operations in the RC4B and BBS. The experiment is executed on a personal computer with Intel Core2 Duo CPU E8400 3.00 GHz, and memory RAM of 4GB. The experimental result is illustrated in the form of graph as shown in Figure 4.1. The result shows that the RC4B needs less execution time to generate the same number of random bits.

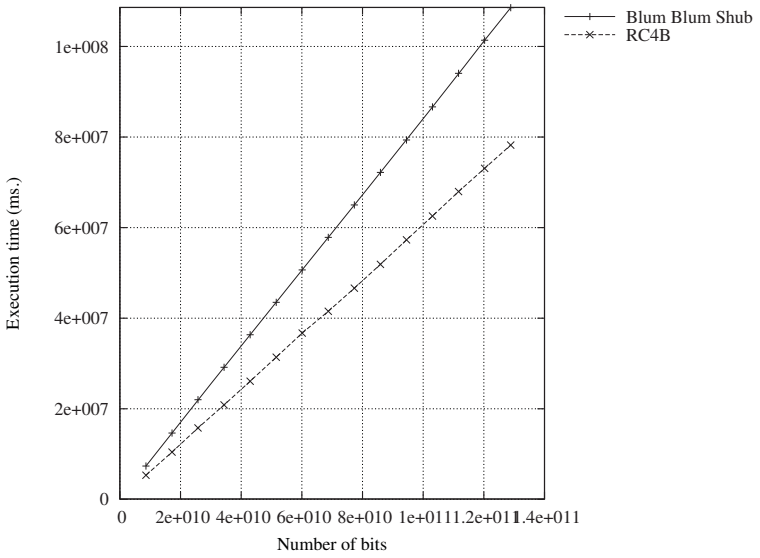


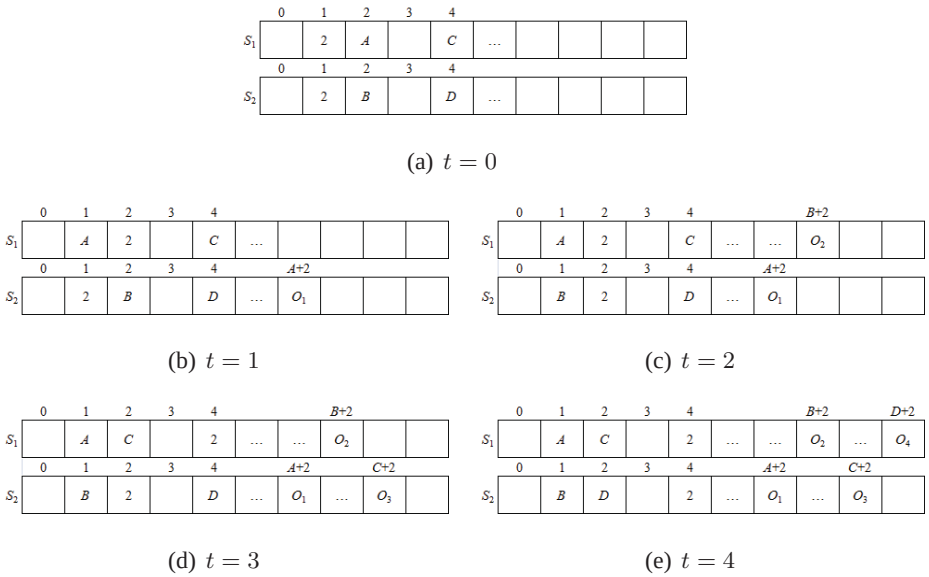
Figure 4.1 Performance comparison between the RC4B and BBS.

4.3.2 Security Analysis

The previous RC4A's biases cannot directly affect the RC4B since each output stream of the RC4B is independently generated from the RC4A's internal state. Each bit stream is generated by BBS algorithm which is computational indistinguishable from real random bits. However, the non-uniformity of the RC4A's internal state may initiate the bias in the RC4B's output. This section aims to investigate the new bias which can threaten the security of RC4A and the RC4B. The result shows that the distinguishing attack on the RC4B requires larger samples than the distinguishing attack on RC4A. Moreover, the number of samples for attacking the RC4B will be increase to infinity by factor of 2^l, where l is the number of RC4B's output bits per round. The related bias is explained in the following theorem.

Theorem IV.2 *Considering RC4A with array's size is $N = 2^8$. Suppose RC4A's KSA can uniformly permute the arrays S_1 and S_2. Assume $S_1[1] = 2$, $S_1[2] \neq 0$ or $S_1[4] \neq 255$, and $S_1[2] \neq 255$ or $S_1[4] \neq 0$. Then, the first and third output bytes are always different. Assume $S_2[1] = 2$, and $S_2[2] \neq 0$. Then, the second and fourth output bytes are always different.*

proof: Figure 4.2 shows the execution of the first two rounds of the RC4A's state tables. From the initial states shown in Figure 4.2(a), the output $O_1 = S_2[A + 2]$, $O_2 = S_1[B + 2]$, $O_3 = S_2[C + 2]$, and $O_4 = S_1[D + 2]$. To prove this theorem, the inequality of O_1 and O_3 is shown and followed by the inequality of O_2 and O_4.



(a) $t = 0$

(b) $t = 1$

(c) $t = 2$

(d) $t = 3$

(e) $t = 4$

Figure 4.2: Array S_1 and S_2 **(a)** at time $t = 0$ **(b)** at time $t = 1$ **(c)** at time $t = 2$ **(d)** at time $t = 3$ **(e)** at time $t = 4$

Assume that $O_1 = O_3$, then $S_2[A + 2] = S_2[C + 2]$. However, $A + 2 \,(\text{mod } 256)$ must not be equivalence to $C + 2 \,(\text{mod } 256)$ because A is not equal to C. The only way that $S_2[A + 2]$ and $S_2[C+2]$ have the same value is the pointers $A+2$ and $C+2$ point to the swapped cells. Considering the following 2 cases:

1. $A + 2 \equiv 1 \,(\text{mod } 256)$ and $C + 2 \equiv 2 \,(\text{mod } 256)$ at time $t = 1$ and $t = 3$. In this case, $A = 255$ and $C = 0$. This situation causes $O_1 = S_2[(255 + 2) \,\text{mod } 256] = S_2[1] = 2$ and $O_3 = S_2[(0 + 2) \,\text{mod } 256] = S_2[2] = 2$.

2. $A + 2 \equiv 2 \,(\text{mod } 256)$ and $C + 2 \equiv 1 \,(\text{mod } 256)$ at time $t = 1$ and $t = 3$. In this case, $A = 0$ and $C = 255$. This situation causes $O_1 = S_2[(0 + 2) \,\text{mod } 256] = S_2[2] = B$ and $O_3 = S_2[(255 + 2) \,\text{mod } 256] = S_2[1] = B$.

From these two cases, the methods to obtain $O_1 = O_3$ are to set $A = 255$ and $C = 0$ or set $A = 0$ and $C = 255$. Consequently, it contradicts to the assumption that $S_1[2] \neq 0$ or $S_1[4] \neq 255$, and $S_1[2] \neq 255$ or $S_1[4] \neq 0$.

The proof of O_2 cannot be equal to O_4 is analogous to the previous proof. Assume that $O_2 = O_4$, then $S_1[B+2] = S_1[D+2]$. As a result, the pointers $B+2$ and $D+2$ point to the S_1's swapped elements and can be divided in to 2 cases:

1. $B + 2 \equiv 2 \,(\text{mod } 256)$ and $D + 2 \equiv 4 \,(\text{mod } 256)$ at time $t = 2$ and $t = 4$. In this case, $B = 0$ and $D = 2$. This situation causes $O_2 = S_1[(0 + 2) \,\text{mod } 256] = S_1[2] = 2$ and

$O_4 = S_1[(2+2) \bmod 256] = S_1[4] = 2$.

2. $B + 2 \equiv 4 \pmod{256}$ and $D + 2 \equiv 2 \pmod{256}$ at time $t = 2$ and $t = 4$. In this case, $B = 2$ and $D = 0$. This situation causes $O_2 = S_1[(2+2) \bmod 256] = S_1[4] = C$ and $O_4 = S_1[(0+2) \bmod 256] = S_1[2] = C$.

The output O_2 and O_4 are equal when $B = 0$ and $D = 2$, or $B = 2$ and $D = 0$. However, it contradicts to the assumption that $S_2[2] \neq 0$. Moreover, $S_2[2]$ and $S_2[4]$ cannot be 2 because $S_2[1]$ is already equal to 2 and all elements in S_1 are different from each other. □

The bias's probability for the first four outputs in Theorem IV.2 can be calculated as shown in the following theorem.

Theorem IV.3 *Considering RC4A with array's size is $N = 2^8$. Let RC4A's KSA can uniformly permute the arrays S_1 and S_2. Assume that all conditions in Theorem IV.2 are satisfied. The probability that $O_1 = O_3$ and $O_2 = O_4$ is $2^{-16} \times \left(1 - 2^{-16.01}\right)$.*

proof: Suppose O_t be the output of the RC4B at time t. Let E_1, E_2, E_3, E_4, and E_5 be the events that $S_1[1] = 2$, $S_1[2] \neq 0$ or $S_1[4] \neq 255$, $S_1[2] \neq 255$ or $S_1[4] \neq 0$, $S_2[1] = 2$, and $S_2[2] \neq 0$. From the assumptions, the probabilities for each event are shown in the following equations.

$$Pr[E_1] = Pr[S_1[1] = 2] = \frac{1}{256} \tag{4.1}$$

$$\begin{aligned} Pr[E_2] &= Pr[S_1[2] \neq 0 \cup S_1[4] \neq 255] \\ &= Pr[S_1[2] \neq 0] + Pr[S_1[4] \neq 255] - Pr[S_1[2] \neq 0 \cap S_1[4] \neq 255] \\ &= \frac{255}{256} + \frac{255}{256} - \frac{255}{256} \times \frac{255}{256} \end{aligned} \tag{4.2}$$

$$\begin{aligned} Pr[E_3] &= Pr[S_1[2] \neq 255 \cup S_1[4] \neq 0] \\ &= Pr[S_1[2] \neq 255] + Pr[S_1[4] \neq 0] - Pr[S_1[2] \neq 255 \cap S_1[4] \neq 0] \\ &= \frac{255}{256} + \frac{255}{256} - \frac{255}{256} \times \frac{255}{256} \end{aligned} \tag{4.3}$$

$$Pr[E_4] = Pr[S_2[1] = 2] = \frac{1}{256} \tag{4.4}$$

$$Pr[E_5] = Pr[S_2[2] \neq 0] = \frac{255}{256} \tag{4.5}$$

Then, the probability that $O_1 = O_3$ and $O_2 = O_4$ can be calculated as follows.

$$\begin{aligned} Pr[O_1 = O_3 \cap O_2 = O_4] &= Pr[O_1 = O_3 \cap O_2 = O_4 | \bigcap_{i=1}^{5} E_i] \times Pr[\bigcap_{i=1}^{5} E_i] \\ &\quad + Pr[O_1 = O_3 \cap O_2 = O_4 | \sim \bigcap_{i=1}^{5} E_i] \times Pr[\sim \bigcap_{i=1}^{5} E_i] \\ &= 0 + 2^{-16} \times \left(1 - \frac{255}{256^3} \times \left(\frac{2 \times 255}{256} - \frac{255^2}{256^2}\right)^2\right) \\ &\approx 2^{-16} \times \left(1 - 2^{-16.01}\right) \end{aligned} \tag{4.6}$$

Therefore, the probability that $O_1 = O_3$ and $O_2 = O_4$ is $2^{-16} \times \left(1 - 2^{-16.01}\right)$ which concludes the prove. □

Furthermore, this bias still appears after dropping the first 256 bytes. The next theorem shows the probability of this bias after dropping the first $256x$ bytes, where $x \geq 2$.

Theorem IV.4 *Suppose all conditions in Theorem IV.3 are satisfied. Let O_t be an RC4A's output byte at time t. The probability that $O_t = O_{t+2}$ and $O_{t+1} = O_{t+3}$ is $2^{-16} \times \left(1 - 2^{-32}\right)$.*

proof: Suppose O_t be the output of RC4B at time t. Suppose the events E_1, E_2, E_3, E_4, and E_5 are the events in Theorem IV.3. Let the events E_6 and E_7 be the event that $j_1^t = 0$ and $j_2^t = 0$. As a result, $Pr[E_6] = Pr[E_7] = \frac{1}{256}$. The probability that $O_t = O_{t+2}$ and $O_{t+1} = O_{t+3}$ can be calculated as follows.

$$\begin{aligned} Pr[O_t = O_{t+2} \cap O_{t+1} = O_{t+3}] &= Pr[O_t = O_{t+2} \cap O_{t+1} = O_{t+3} | \bigcap_{i=1}^{7} E_i] \times Pr[\bigcap_{i=1}^{7} E_i] + \\ &\quad Pr[O_t = O_{t+2} \cap O_{t+1} = O_{t+3} | \sim \bigcap_{i=1}^{7} E_i] \times Pr[\sim \bigcap_{i=1}^{7} E_i] \\ &= 0 + 2^{-16} \times \left(1 - \frac{255}{256^5} \times \left(\frac{2 \times 255}{256} - \frac{255^2}{256^2}\right)^2\right) \\ &\approx 2^{-16} \times \left(1 - 2^{-32}\right) \end{aligned} \tag{4.7}$$

Therefore, the probability that $O_t = O_{t+2}$ and $O_{t+1} = O_{t+3}$ is $2^{-16} \times \left(1 - 2^{-32}\right)$. □

The method for estimating the number of samples that is used to distinguish stream cipher's distribution from the uniform distribution is described and proved in paper [61]. This research will use this theorem for calculating the number of samples used for distinguishing the RC4A's random bit from real random bits.

Theorem IV.5 *Let e be an event and let p and $p(1+q)$ be probabilities of the event e occurs in the distribution X and the distribution Y, respectively. The amount of samples used to distinguish X from Y with non-negligible probability of success is $O\left(\frac{1}{pq^2}\right)$.*

Theorem IV.5 explains the amount of samples used for distinguishing the distribution X from the distribution Y. In this case, the distribution X and Y are uniform distribution and the RC4A's output distribution. The event e is the event that the first and third random bytes are equal and the second and fourth random bytes are equal. The probability p and q are equal to 2^{-16} and $2^{-16.01}$. Therefore, the number of samples used to distinguish the bias in Theorem IV.3 is $2^{48.02}$. In case of dropping first $256x$ bytes, the probability q is equal to 2^{-32}. The number of samples used to distinguish the bias in Theorem IV.4 is 2^{80}. Note that, this value is based on the assumption that RC4A's internal states are uniformly permuted by KSA.

The bias proposed in Theorem IV.3 forces the RC4B to produce the same connected random streams when using the same co-prime, generated by **GenCoprime** algorithm. The following theorem will calculate the probability that the seed from **GenCoprime** algorithm can activate BBS to generate the same sequence.

Theorem IV.6 *Let p and q be the Blum primes. Let $n = p \times q$ be a Blum integer. Assume that **GenCoprime** algorithm can uniformly choose $s \in \mathbb{Z}_n^*$. The probability that the **GenCoprime**(s, n) chooses the seed which can set BBS to generate the same sequence at time t is defined as a function*

$$\rho(n,t) = 1 - \prod_{i=1}^{t} \frac{\phi(n) - 4(i-1)}{\phi(n)}$$

where $\phi(n)$ denotes Euler's totient function.

proof: Let Q_t is the probability that **GenCoprime**(s, n) chooses the different seed at time t. From quadratic residuosity properties, the number of seeds which can be used to generate the different bit streams is $\frac{\phi(n)}{4}$. At time

$$\begin{aligned}
t = 1, \;\; Q_1 &= \frac{\phi(n)/4}{\phi(n)/4} \\
t = 2, \;\; Q_2 &= \frac{\phi(n)/4}{\phi(n)/4} \times \frac{\phi(n)/4 - 1}{\phi(n)/4} \\
t = 3, \;\; Q_3 &= \frac{\phi(n)/4}{\phi(n)/4} \times \frac{\phi(n)/4 - 1}{\phi(n)/4} \times \frac{\phi(n)/4 - 2}{\phi(n)/4}
\end{aligned}$$

Therefore, at time t,

$$Q_t = \prod_{i=1}^{t} \frac{\phi(n) - 4(i-1)}{\phi(n)}$$

As a result,

$$\rho(n,t) = 1 - Q_t = 1 - \prod_{i=1}^{t} \frac{\phi(n) - 4(i-1)}{\phi(n)}$$

□

The probability that the RC4B generates the same consecutive sequences is calculated in the next theorem.

Theorem IV.7 *Considering the RC4B with the array's size is $N = 2^8$. Suppose the RC4B's KSA can uniformly permute the arrays P and Q and let **GenCoprime** method can uniformly select the co-prime from $\mathbb{Z}_n^*$. Assume that all conditions in Theorem IV.2 are satisfied. The probability that the first and second streams of the RC4B are equal is $\frac{1}{2^l} \times \left(1 - 2^{-16.01} \times \frac{\phi(n) - 4}{\phi(n)}\right)$, where l is the number of output bits per round and n is the RC4B's Blum integer that has minimum value of $\phi(n)$.*

proof: Suppose O_t be the output of the RC4B at time t. Let s_t be the co-prime chosen by **GenCoprime** method at time t. Let E_1, E_2, E_3, E_4, E_5, and E_6 be the events that $P[1].getIndex() = 2$, $P[2].getIndex() \neq 0$ or $P[4].getIndex() \neq 255$, $P[2].getIndex() \neq 255$ or $P[4].getIndex() \neq 0$, $Q[1].getIndex() = 2$, $Q[2].getIndex() \neq 0$, and $s_1 \neq s_2$. The event $O_1 = O_2$ occurs with probability

$$\begin{aligned}
Pr[O_1 = O_2] &= Pr[O_1 = O_2 | \bigcap_{i=1}^{6} E_i] \times Pr[\bigcap_{i=1}^{6} E_i] + Pr[O_1 = O_2 | \sim \bigcap_{i=1}^{6} E_i] \times Pr[\sim \bigcap_{i=1}^{6} E_i] \\
&= 0 + \frac{1}{2^l} \times \left(1 - \frac{255}{256^3} \times \left(\frac{2 \times 255}{256} - \frac{255^2}{256^2}\right)^2 \times \left(\frac{\phi(n) - 4}{\phi(n)}\right)\right) \\
&\approx \frac{1}{2^l} \times \left(1 - 2^{-16.01} \times \frac{\phi(n) - 4}{\phi(n)}\right) \qquad (4.8)
\end{aligned}$$

Note that, the probability of the event E_6 can be calculated from $1 - \rho(n, 2)$. This is the lower bound probability since n has the minimum value of Euler's totient function. □

Next theorem will show the number samples used for distinguishing the RC4B's bit stream from real random bits.

Theorem IV.8 *Considering the RC4B with array's size is $N = 2^8$. Suppose the RC4B's KSA can uniformly permute the arrays P and Q and **GenCoprime** method can uniformly choose co-prime from $\mathbb{Z}_n^*$. The number of samples used for distinguishing the RC4B's bit stream is at most $2^{32.02+l}$.*

proof: From Theorem IV.5, the number of samples used for distinguishing the distribution X from distribution Y on the event e is $O\left(\frac{1}{pq^2}\right)$. In this case, p and q are $\frac{1}{2^l}$ and $2^{-16.01} \times \frac{\phi(n) - 4}{\phi(n)}$. As

a result, the number of samples is $O\left(2^{32.02+l} \times \left(\frac{\phi(n)}{\phi(n)-4}\right)^2\right) \approx 2^{32.02+l}$ when $\phi(n)$ has large value. $\square$

From Theorem IV.8, the number of samples is exponentially growth by the term of 2^l. Practically, the implementation of traditional BBS is usually used $n \approx 10^{160}$ but the RC4B can use the smaller Blum integers. If the two smallest chosen Blum primes have a value nearly to 10^{10} and the number of random bits per round is 200, it will increase the number of samples approximate to 2^{232} samples. Moreover, Theorem IV.6 can be used to determine the optimal Blum prime's size in array P and Q for avoiding the proposed bias. The method for determining the optimal Blum prime's size is shown in Algorithm IV.5.

Algorithm IV.5 Algorithm for Determining the optimal Blum primes

Input: n is the Blum integer which has minimum $\phi(n)$, and
X is the amount of required random bits.

Output: accept or reject the Blum integer.

$r = X/\lambda(\lambda(n))$, $p = 1.0$

for $i = 1 \rightarrow r$ **do**

$p = p \times (1 - 4(i-1)/\phi(n))$

end for

if $p > 0.5$ **then**

return accept.

else

return reject.

end if

4.4 Results and Discussions

The RC4B's output sequences are examined by NIST Statistical Test Suite. The statistical tests have been executed over 100 samples. The results from NIST consists of two parts, the proportion of sequences that pass the statistical test, and the uniform distribution of P-value. The results are shown in Table 4.1. The first column represents the name of statistical test, the second column represents the proportion of sequence that pass the statistical test, and the third column represents the P-value that arises via the application of a chi-square test. The minimum pass rate for each statistical test with the exception of the random excursion (variant) test is approximately to 96 when the number of

Table 4.1 The Statistical Tests on RC4B by NIST Statistical Test Suite

Statistical Test	Proportion	P-Value
Frequency	99/100	0.494392
BlockFrequency	100/100	0.494392
CumulativeSums	99/100	0.289667
CumulativeSums	99/100	0.366918
Runs	100/100	0.289667
LongestRun	100/100	0.678686
Rank	99/100	0.983453
FFT	100/100	0.213309
NonOverlappingTemplate	100/100	0.534146
NonOverlappingTemplate	98/100	0.137282
NonOverlappingTemplate	99/100	0.719747
NonOverlappingTemplate	99/100	0.319084
OverlappingTemplate	96/100	0.000296
Universal	99/100	0.494392
ApproximateEntropy	99/100	0.455937
RandomExcursions	81/83	0.448892
RandomExcursions	82/83	0.572333
RandomExcursions	83/83	0.969045
RandomExcursions	83/83	0.381687
RandomExcursionsVariant	83/83	0.381687
RandomExcursionsVariant	83/83	0.598138
RandomExcursionsVariant	83/83	0.902994
RandomExcursionsVariant	83/83	0.823278
Serial	99/100	0.657933
Serial	99/100	0.262249
LinearComplexity	99/100	0.334538

samples is 100. The minimum pass rate for the random excursion (variant) test is approximately to 81 when the number of samples is 83. If P-value ≥ 0.0001, then the sequences can be considered to be uniformly distributed. This table shows that RC4B's output sequences pass all statistical tests.

The RC4B obtains the advantages in speed of RC4A and the security of BBS. It also resists the previous attacks on RC4A. The RC4A's critical bias was proposed in [55]. This bias affects the RC4B in choosing the same Blum prime at time $t = 1$ and time $t = 3$ with a probability is not equal to 2^{-n}, where n is a size of the RC4B's array. It will not affect to the RC4B's output stream because the output stream is generated via BBS which is independent from the shuffle process. Another distinguishing attack was proposed in [54]. Like the previous bias, it affects to the probability that the same Blum prime is chosen at time t and time $t + 2$ when i is even but it does not directly influence to the output stream.

This research further studies the correlated bias that has impacts on both RC4A and the RC4B. The new bias causes the non-uniform distribution on RC4A's first four output bytes. The proof shows that the new bias can be used to distinguish the RC4A's bit stream with $2^{48.02}$ samples. On the other hand, the same bias can cause only a tiny defect to the RC4B. Thus, to create the distinguisher for the RC4B on this bias, $2^{32.02+l}$ samples are required which are much more than the samples required for distinguishing RC4A because the number of samples tend to increase exponentially by factor of 2^l.

The RC4B also has the faster execution time comparing to BBS as shown in Section 4.3.1. The performance is very important issue in the encryption/decryption process. The BBS is failed to use as a stream cipher because it has low performance. The RC4B reduces the BBS's computation complexity to obtain higher performance and still reserve the BBS's security as much as possible.

4.5 Conclusions

This chapter presents the hybrid algorithm based on RC4A and BBS. The RC4B's designed principles are to enhance the security of the stream cipher and to reduce the calculation complexity of the CSPRG. The RC4A's previous attacks are hopeless when they confront with the RC4B because each output stream of the RC4B is generated independent from RC4A's internal state. It also decreases the correlation between external and internal states. Moreover, this research described the new bias for RC4A which can affect to the RC4B's security. The proofs show that to distinguish the RC4B, the large amounts of samples are required and they are extended by the factor of 2^l, which is exponentially increasing function. The RC4B's execution time is less than the BBS's exeucution time as shown in Theorem IV.1 and the experimental result. For conclusion, the RC4B gains much more security over

RC4A and it needs less execution time compare to BBS. The expected outcome of this research is that it can be used as a prototype of the new era of the stream cipher which can conserve the performance and obtain more security from the CSPRG family.

CHAPTER V

PROVABLE SECURE PSEUDORANDOM NUMBER GENERATOR BASED ON LEARNING WITH ERRORS PROBLEM

This chapter proposes the provable secure pseudorandom generator based on the computational difficulty of Learning with Errors problem, the LWE-based PRG. The proposed PRG is proven that it can resist against the attack by quantum computers. Formally, the problem of distinguishing the random bits from LWE-based PRG is proven that it is not easier than solving Regev's Learning with Errors (Regev's LWE) problem, which is quantumly as hard as solving the worst-case lattices problems.

This chapter is organized into 4 sections. In Section 5.1, some mathematical notations are defined. In Section 5.2, the construction of PRG and its security proof are presented. The advantages of the proposed PRG over the previous cryptographic secure PRGs and the optimization of the security parameters are explained in Section 5.3. The conclusions of this chapter is shown in Section 5.4.

5.1 Notations

In this chapter, the normal, bold, and capital bold letters like $x, \mathbf{x}, \mathbf{X}$ denote the single variables, vectors, and matrices, respectively. The symbol $x \xleftarrow{\$} X$ denotes that x is uniform sampled from the set X. Let $\mathcal{X}$ be the distribution, the symbol $x \leftarrow \mathcal{X}$ denotes the sampling process of the value x from the distribution $\mathcal{X}$. For any integer $q > 1$, the set of integers modulo q is denoted by $\mathbb{Z}_q$.

5.2 LWE-based Pseudorandom Number Generator

This section presents the construction of LWE-based PRG. Then, the security proof of the proposed method is shown in the end of this section.

5.2.1 Constructions

This research considers only the special case of LWE, Regev's LWE problem [29]. Recall that Regev's LWE problem is LWE when the modulus is prime and the noise distribution is discrete Gaussian distribution. Specifically, let n be the security parameter and $P(n)$ be the positive polynomial

of n, Regev's LWE problem uses the modulus q to be prime number such that $q \leq P(n)$ and the noise distribution $\mathcal{X} = \mathcal{N}\left(0, \alpha^2 q^2/2\pi\right)$ rounded to the nearest integer and modulo q, for any real $\alpha \in (0,1)$ and $\alpha q > \sqrt{n}$. The concept of the proposed generator is that, given the tuple $(\mathbf{A}, \mathbf{x}, \mathbf{e})$ where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{2n \times n}$, $\mathbf{x} \xleftarrow{\$} \mathbb{Z}_q^n$, and $\mathbf{e} \leftarrow \mathcal{X}^{2n}$, it is easy to compute the pseudorandom outputs. In contrast, the problem of distinguishing the outputs from the LWE-based PRG is at least as hard as Regev's LWE problem in decision version.

The input parameters to the LWE-based PRG are the matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{2n \times n}$, the secret vector $\mathbf{x} \xleftarrow{\$} \mathbb{Z}_q^n$, and the noise vector $\mathbf{e} \leftarrow \mathcal{N}^{2n}\left(0, \alpha^2 q^2/2\pi\right)$. For any indices $i, j \in \{1, 2, \ldots, n\}$, the symbol $\mathbf{x}[i]$ denotes the i-th element of $\mathbf{x}$ and the symbol $\mathbf{x}[i \ldots j]$ denotes the subvector of $\mathbf{x}$ from i-th element to j-th element. The construction is shown in the Algorithm V.1.

Algorithm V.1 LWE-based Pseudorandom Generator

Input: $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{2n \times n}, \mathbf{x} \xleftarrow{\$} \mathbb{Z}_q^n$, and $\mathbf{e} \leftarrow \mathcal{N}^{2n}\left(0, \alpha^2 q^2/2\pi\right)$.

Output: The pseudorandom numbers.

$\mathbf{x}_0 = \mathbf{x}$

for $i = 1$ to l **do**

$\mathbf{y} = (\mathbf{A}\mathbf{x}_{i-1} + \mathbf{e}) \mod q$

$\mathbf{x}_i = \mathbf{y}[1, \ldots, n], \mathbf{z}_i = \mathbf{y}[n+1 \ldots 2n]$

end for

Output $\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_l, \mathbf{A}$

The input's length to the LWE-based PRG is $s(n,q) = \left(2n^2 + 3n\right)\|q\|$ bits and the output's length is $t(n,q,l) = \left(2n^2 + ln\right)\|q\|$ bits, where $\|q\|$ is the size of modulus q. The expansion factor of LWE-based PRG is $t(n,q,l) - s(n,q) = n(l-3)\|q\|$. Therefore, the number of iteration $l > 3$ is selected to obtain the expansion provided.

5.2.2 Security Analysis

The security proof of the LWE-based PRG is relied on the following assumption.

Assumption V.1 *Let $n \in \mathbb{Z}^+$ be the security parameter, $P(n)$ be the positive polynomial of n. Assume that $q \leq P(n)$ is a prime, $\alpha \in (0,1)$ is a real number, and the noise distribution $\mathcal{X}$ is the Gaussian noise distribution such that $\mathcal{X} = \mathcal{N}\left(0, \alpha^2 q^2/2\pi\right)$ and $\alpha q > \sqrt{n}$. Assume $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{2n \times n}$ is the public matrix, $\mathbf{x} \xleftarrow{\$} \mathbf{Z}_q^n$ is the uniform secret vector, $\mathbf{e} \leftarrow \mathcal{X}^n$ is the secret noise vector, and D is*

the PPT algorithm for solving Regev's LWE problem. Then,

$$\left|\Pr\left[D\left(\mathbf{A},\left(\mathbf{Ax}+\mathbf{e}\right)\bmod q\right)=1\right]-\Pr\left[D\left(\mathbf{A},\mathbf{y}\right)=1\right]\right|<\frac{1}{Q\left(n\right)} \tag{5.1}$$

where $\mathbf{y}$ *is the uniformly distributed vector over* $\mathbb{Z}_2^{2n}$ *and* $Q\left(n\right)$ *is the positive polynomial of* n*.*

Note that the security proof only considers the computational indistinguishability of the output vectors $\mathbf{z}_1,\mathbf{z}_2,...,\mathbf{z}_l$ and omits the analysis on the output matrix $\mathbf{A}$ since the matrix $\mathbf{A}$ is assumed to be uniformly distributed over $\mathbb{Z}_q^{2n\times n}$. To prove the security of the LWE-based PRG, the problem of distinguishing Regev's LWE samples from the uniform distribution is reducible to the problem of distinguishing the LWE-based PRG's outputs from the uniform distribution.

Theorem V.1 *If Assumption V.1 holds, then LWE-based PRG is a pseudorandom generator.*

proof: Assume that, for the sake of contraposition, the LWE-based PRG is not pseudorandom generator, then, Assumption V.1 does not hold. Consequently, there exist the PPT algorithm D' for distinguishing the LWE-based PRG's outputs from uniform random sequences. Formally, let $\mathcal{G}$ denotes the LWE-based PRG, there exist the PPT adversary algorithm D' such that

$$\left|\Pr\left[D'\left(\mathbf{A},\mathcal{G}\left(\mathbf{A},\mathbf{x},\mathbf{e}\right)\right)=1:\mathbf{x}\overset{\$}{\leftarrow}\mathbb{Z}_q^n\right]-\Pr\left[D'\left(\mathbf{A},\mathbf{y}\right)=1:\mathbf{y}\overset{\$}{\leftarrow}\mathbb{Z}_q^{l\times n}\right]\right|>\frac{1}{R\left(n\right)} \tag{5.2}$$

where $R\left(n\right)$ is the positive polynomial of n. This proof is shown by constructing the PPT algorithm D, using D' as a subroutine, for solving Regev's LWE with advantage better than $\frac{1}{Q(n)}$. Let $\alpha^k\in\mathbb{Z}_q^{kn}$ be the concatenation of the chosen k samples, each sample $\alpha_i^k\in\mathbb{Z}_q^n$ where $1\leq i\leq k$. The function $\mathrm{prefix}_j\left(\alpha^k\right)$ denotes the first j samples of string α^k. The function $\mathrm{suffix}_j\left(\alpha^k\right)$ denotes the last j samples of string α^k. The first $j+1$ outputs from $\mathcal{G}\left(\mathbf{A},\mathbf{x},\mathbf{e}\right)$ can be written in terms of prefix and suffix as follow

$$\mathrm{prefix}_{j+1}(\mathcal{G}\left(\mathbf{A},\mathbf{x},\mathbf{e}\right))=\mathrm{suffix}_1((\mathbf{Ax}+\mathbf{e})\bmod q)\cdot\mathrm{prefix}_j(\mathcal{G}\left(\mathbf{A},\mathrm{prefix}_1((\mathbf{Ax}+\mathbf{e})\bmod q),\mathbf{e}\right)). \tag{5.3}$$

For any $l,k\in\mathbb{Z}^+$ and $k<l$, let f_k^l be the function for performing the computation on the input string, α^2, as follow:

$$\begin{aligned}f_k^l\left(\alpha^2\right)&=\alpha_2^2\cdot\mathrm{prefix}_{l-k-1}\left(\mathcal{G}\left(\mathbf{A},\alpha_1^2,\mathbf{e}\right)\right)\\&=\mathrm{suffix}_1\left(\alpha^2\right)\cdot\mathrm{prefix}_{l-k-1}\left(\mathcal{G}\left(\mathbf{A},\mathrm{prefix}_1\left(\alpha^2\right),\mathbf{e}\right)\right)\end{aligned} \tag{5.4}$$

The algorithm D for solving Regev's LWE problem with non-negligible advantage is shown in Algorithm V.2.

Algorithm V.2 Construction of D

Input: $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{2n\times n}, \alpha^2 \in \mathbb{Z}_q^{2n}$

Output: 0 or 1

$k \xleftarrow{\$} \{0, 1, 2, \ldots, l-1\}$

$\beta \xleftarrow{\$} \mathbb{Z}_q^{kn}$

return $D'\left(\mathbf{A}, \beta \cdot f_k^l\left(\alpha^2\right)\right)$

The advantage of Algorithm V.2 is proven as follow. Let H_k^l be the hybrid random variable represents the concatenation between the uniformly distributed samples and the outputs from $\mathcal{G}$. Specifically, the first k samples of H_k^l are uniformly sampling from the set $\mathbb{Z}_q^n$ and the rest $l-k$ samples of H_k^l are the first $l-k$ outputs of $\mathcal{G}$. Let $\mathcal{U}_k$ be the uniformly distributed sample over $\mathbb{Z}_q^{kn}$, where $0 \leq k \leq l$. Consider the following four cases:

$$
\begin{aligned}
H_0^l &= \mathcal{U}_0 \cdot \mathrm{prefix}_l\left(\mathcal{G}\left(\mathbf{A}, \mathcal{U}_1, \mathbf{e}\right)\right) = \mathcal{G}\left(\mathbf{A}, \mathbf{x}, \mathbf{e}\right) \\
H_k^l &= \mathcal{U}_k \cdot \mathrm{prefix}_{l-k}\left(\mathcal{G}\left(\mathbf{A}, \mathcal{U}_1, \mathbf{e}\right)\right) \qquad (5.5) \\
H_{k+1}^l &= \mathcal{U}_{k+1} \cdot \mathrm{prefix}_{l-k-1}\left(\mathcal{G}\left(\mathbf{A}, \mathcal{U}_1, \mathbf{e}\right)\right) \qquad (5.6) \\
H_l^l &= \mathcal{U}_l \cdot \mathrm{prefix}_{l-l}\left(\mathcal{G}\left(\mathcal{U}_1, \mathbf{x}, \mathbf{e}\right)\right) = \mathcal{U}_l
\end{aligned}
$$

The hybrid random variables H_0^l and H_l^l are distributed according to $\mathcal{G}\left(\mathbf{A}, \mathbf{x}, \mathbf{e}\right)$ and $\mathcal{U}_l$, respectively. Thus, the ability to distinguish LWE-based PRG output from the uniform sequence can be translated to the ability to distinguish H_0^l from H_l^l. The advantage of distinguisher D' in Equation 5.2 can be revised as follow

$$
\left|\Pr\left[D'\left(\mathbf{A}, H_0^l\right) = 1\right] - \Pr\left[D'\left(\mathbf{A}, H_l^l\right) = 1\right]\right| > \frac{1}{R\left(n\right)}. \qquad (5.7)
$$

The distribution of hybrid random variables H_k^l and H_{k+1}^l are shown in the following lemma.

Lemma V.1 *For any $0 \leq k \leq l$, let $\mathcal{U}_k$ be the uniformly distributed sample over $\mathbb{Z}_q^{kn}$. Let $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{2n\times n}$ be the uniform sampling matrix over $\mathbb{Z}_q^{2n\times n}$ and $\mathbf{e} \leftarrow \mathcal{N}^{2n}\left(0, \alpha^2 q^2/2\pi\right)$ be the noise vector. Then,*

1. *The hybrid random variable H_k^l is distributed identically to $\mathcal{U}_k \cdot f_k^l\left(\left(\mathbf{A}\mathcal{U}_1 + \mathbf{e}\right) \bmod q\right)$.*
2. *The hybrid random variable H_{k+1}^l is distributed identically to $\mathcal{U}_k \cdot f_k^l\left(\mathcal{U}_2\right)$.*

proof of Lemma V.1: The hybrid random variables H_k^l and H_{k+1}^l can be revised in term of f_k^l by combining Equation 5.3, 5.4, 5.5, and 5.6.

$$\begin{aligned} H_k^l &= \mathcal{U}_k \cdot \mathrm{prefix}_{(l-k-1)+1}\left(\mathcal{G}\left(\mathbf{A}, \mathcal{U}_1, \mathbf{e}\right)\right) \\ &= \mathcal{U}_k \cdot \mathrm{suffix}_1\left(\left(\mathbf{A}\mathcal{U}_1 + \mathbf{e}\right) \bmod q\right) \cdot \mathrm{prefix}_{l-k-1}\left(\mathcal{G}\left(\mathbf{A}, \mathrm{prefix}_1\left(\left(\mathbf{A}\mathcal{U}_1 + \mathbf{e}\right) \bmod q\right), \mathbf{e}\right)\right) \\ &= \mathcal{U}_k \cdot f_k^l\left(\left(\mathbf{A}\mathcal{U}_1 + \mathbf{e}\right) \bmod q\right) \qquad (5.8) \\ H_{k+1}^l &= \mathcal{U}_{k+1} \cdot \mathrm{suffix}_{l-k-1}\left(\mathcal{G}\left(\mathbf{A}, \mathcal{U}_1, \mathbf{e}\right)\right) \\ &= \mathcal{U}_k \cdot \mathrm{suffix}_1\left(\mathcal{U}_2\right) \cdot \mathrm{prefix}_{l-k-1}\left(\mathcal{G}\left(\mathbf{A}, \mathrm{prefix}_1\left(\mathcal{U}_2\right), \mathbf{e}\right)\right) \\ &= \mathcal{U}_k \cdot f_k^l\left(\mathcal{U}_2\right) \qquad (5.9) \end{aligned}$$

From Equation 5.8 and 5.9, H_k^l and H_{k+1}^l are distributed identically to $\mathcal{U}_k \cdot f_k^l\left(\left(\mathbf{A}\mathcal{U}_1 + \mathbf{e}\right) \bmod q\right)$ and $\mathcal{U}_k \cdot f_k^l\left(\mathcal{U}_2\right)$. □

Lemma V.1 implies that the ability to distinguish $\mathcal{U}_k \cdot f_k^l\left(\left(\mathbf{A}\mathcal{U}_1 + \mathbf{e}\right) \bmod q\right)$ from $\mathcal{U}_k \cdot f_k^l\left(\mathcal{U}_2\right)$ is transformed to the ability to distinguish hybrid random variable H_k^l from H_{k+1}^l. The probability that the algorithm D, for solving Regev's LWE, replies 1 on input α^2 can be determined in term of hybrid variables H_k^l and H_{k+1}^l as follow.

$$\begin{aligned} \Pr\left[D\left(\mathbf{A}, \left(\mathbf{A}\mathbf{x} + \mathbf{e}\right) \bmod q\right) = 1\right] &= \frac{1}{l}\sum_{k=0}^{l-1} \Pr\left[D'\left(\mathbf{A}, \mathcal{U}_k \cdot f_k^l\left(\left(\mathbf{A}\mathbf{x} + \mathbf{e}\right) \bmod q\right)\right) = 1\right] \\ &= \frac{1}{l}\sum_{k=0}^{l-1} \Pr\left[D'\left(\mathbf{A}, H_k^l\right) = 1\right] \qquad (5.10) \\ \Pr\left[D\left(\mathbf{A}, \mathcal{U}_2\right) = 1\right] &= \frac{1}{l}\sum_{k=0}^{l-1} \Pr\left[D'\left(\mathbf{A}, \mathcal{U}_k \cdot f_k^l\left(\mathcal{U}_2\right)\right) = 1\right] \\ &= \frac{1}{l}\sum_{k=0}^{l-1} \Pr\left[D'\left(\mathbf{A}, H_{k+1}^l\right) = 1\right] \qquad (5.11) \end{aligned}$$

Assume $\mathrm{Adv}\left(n\right)$ is the advantage of D such that

$$\mathrm{Adv}\left(n\right) = \left|\Pr\left[D\left(\mathbf{A}, \left(\mathbf{A}\mathbf{x} + \mathbf{e}\right) \bmod q\right) = 1\right] - \Pr\left[D\left(\mathbf{A}, \mathcal{U}_2\right) = 1\right]\right|.$$

From Equation 5.7, 5.10, and 5.11, $\mathrm{Adv}\left(n\right)$ can be calculated as follow.

$$\begin{aligned} \mathrm{Adv}\left(n\right) &= \frac{1}{l}\sum_{k=0}^{l-1} \left|\Pr\left[D'\left(\mathbf{A}, H_k^l\right) = 1\right] - \Pr\left[D'\left(\mathbf{A}, H_{k+1}^l\right) = 1\right]\right| \\ &= \frac{1}{l}\left|\Pr\left[D'\left(\mathbf{A}, H_0^l\right) = 1\right] - \Pr\left[D'\left(\mathbf{A}, H_l^l\right) = 1\right]\right| \\ &> \frac{1}{lR\left(n\right)} \end{aligned}$$

Therefore, the algorithm D for solving Regev's LWE problem, constructed from distinguisher of LWE-based PRG, has an advantage larger than $\frac{1}{Q(n)}$, where $Q(n) = lR(n)$, which appears to contradict Assumption V.1. □

5.3 Results and Discussions

The LWE-based PRG's output sequences are examined by NIST Statistical Test Suite. The statistical tests have been executed over 100 samples. The results from NIST consists of two parts, the proportion of sequences that pass the statistical test and the uniform distribution of P-value. The results are shown in Table 5.1. The first column represents the name of statistical test, the second column represents the proportion of sequence that pass the statistical test, and the third column represents the P-value that arises via the application of a chi-square test. The minimum pass rate for each statistical test with the exception of the random excursion (variant) test is approximately to 96 when the number of samples is 100. The minimum pass rate for the random excursion (variant) test is approximately to 86 when the number of samples is 88. If P-value ≥ 0.0001, then the sequences can be considered to be uniformly distributed. This table shows that LWE-based PRG's output sequences pass all statistical tests.

The LWE-based PRG gains many advantages over the previous cryptographic secure PRGs. First, it is computational indistinguishable by any quantum algorithms. The problem of distinguishing the LWE-based PRG's outputs can be proven that it at least as hard as solving Regev's LWE problem. This problem is known to be as hard as worst-case lattice problems, which are believed to be exponentially hard, even against quantum computers. Differently, the traditional provable secure PRGs are usually constructed on Factoring assumption or Discrete Logarithm assumption, which can be solved in polynomial time when using quantum algorithms.

Second, it contains several parallel steps, that are very important when implement in both software and hardware. The prior provable secure PRGs mostly used modular exponentiation operation to compute the random bits. It is time consuming for large operands and containing less parallel steps.

Third, the hardness of Regev's LWE assumption depends on the secret vector's dimension and the modulus. There are many proofs show that when the dimension and modulus are large enough, it cannot compute the answer of LWE in an appropriate time. This feature makes the LWE-based PRG's security based on many parameters while the antecedent PRGs's security parameter is only the modulus's size.

Fourth, the implementation of the LWE-based PRG in 32-bits machine is very simple. Only

Table 5.1 The Statistical Tests on LWE-based PRG by NIST Statistical Test Suite

Statistical Test	Proportion	P-Value
Frequency	100/100	0.383827
BlockFrequency	100/100	0.455937
CumulativeSums	100/100	0.494392
CumulativeSums	100/100	0.350485
Runs	99/100	0.191687
LongestRun	98/100	0.262249
Rank	98/100	0.262249
FFT	99/100	0.935716
NonOverlappingTemplate	98/100	0.122325
NonOverlappingTemplate	99/100	0.062821
NonOverlappingTemplate	99/100	0.883171
NonOverlappingTemplate	98/100	0.678686
OverlappingTemplate	96/100	0.102526
Universal	98/100	0.085587
ApproximateEntropy	99/100	0.037566
RandomExcursions	88/88	0.057146
RandomExcursions	86/88	0.002971
RandomExcursions	86/88	0.559523
RandomExcursions	86/88	0.689019
RandomExcursionsVariant	87/88	0.392456
RandomExcursionsVariant	87/88	0.739918
RandomExcursionsVariant	87/88	0.350485
RandomExcursionsVariant	87/88	0.980883
Serial	100/100	0.897763
Serial	99/100	0.383827
LinearComplexity	100/100	0.534146

ANSI C compiler with standard library can be used to implement the LWE-based PRG which can satisfy the basic requirement of security parameters. For example, using the vector's dimension $n = 300$ and the modulus q is the 32-bits prime number. The PRGs that based on factoring assumption always require another library, such as GMP or NTL, for storing and operating on 1024 bits modulus.

However, the LWE-based PRG requires more randomness than the previous PRGs. This randomness makes the LWE-based PRG is inflexible in the environment that the cost of generating the randomness is expensive, such as laptops or mobile devices. Ring-LWE assumption can be used instead of the Regev's LWE assumption for optimizing randomness and execution time. This simple adaptation can be done by changing the matrix $\mathbf{A}$ to circulant structure. This approach is similar to the design of NTRU public key cryptosystem [62]. There are researches prove that Ring-LWE is quantumly as hard as solving the worst-case ideal lattices problems [63, 64]. Moreover, the memory and the number of operations per bit are reduced when using Fast Fourier Transform for performing matrix-vector multiplication on the ring structure.

5.4 Conclusions

This chapter proposed the provable secure PRG which can resist against the quantum distinguishing attack. The hardness of distinguishing the random bits from traditional cryptographic secure PRGs are based on Factoring and Discrete Logarithm assumptions. Unfortunately, these problems can be efficiently solved by the quantum algorithm. The construction of the proposed PRG is based on the hardness of Regev's LWE. There is no quantum algorithm for efficiently solving Regev's LWE. The problem of solving Regev's LWE problem is reducible to the problem of distinguishing the LWE-based PRG's outputs. This proof ensures that the problem of distinguishing the random bits from the LWE-based PRG is at least as hard as Regev's LWE. Therefore, the LWE-based PRG is resisting against the quantum distinguishing attack.

CHAPTER VI

THE HARDNESS OF LEARNING PARITY WITH NOISE ASSUMPTION FOR NON-UNIFORM SECRET KEY

Standard LPN usually assumes that the secret vector is securely kept and uniformly distributed. The hardness of LPN when the secret vector is sampled from arbitrary distribution with sufficient high min-entropy still unclear [65]. Unlike LPN, LWE with secret key $\mathbf{s} \in \mathbb{Z}_q^n$ has min-entropy k is as hard as standard LWE with secure parameter at most $(k - \omega(\log n)) / \log q$ [34]. The proof of LWE implies that it requires the modulus q to be at least super-polynomial of n. Therefore, it means nothing about the case of LPN where $q = 2$.

This research proves one of the LPN's open questions, the hardness of LPN when the secret vector is drawn from general distributions of high min-entropy, denoted as H_∞LPN. More specifically, this research shows that the problem of LPN when the secret vector in $\mathbb{Z}_2^n$ having min-entropy k is as hard as standard LPN with secret vector in $\mathbb{Z}_2^k$. This work takes the different approach from the original paper of LWE. The H_∞LPN is reducible to Subspace LPN, which is known to be as hard as LPN [66]. Furthermore, the hardness of H_∞LPN is used to construct the cryptographic application which is robust to the weak secret key.

This chapter is organized into 4 sections. In Section 6.1, the related definitions about Standard LPN and Subspace LPN is defined. The hardness of H_∞LPN is proven in Section 6.2. In Section 6.3, some discussions about the hardness of H_∞LPN and the relation to the hardness of LWE when the secret vector in $\mathbb{Z}_q^n$ has min-entropy k are described. This research also presents the symmetric-key encryption scheme, based on the hardness of LPN, which is secure even if the secret key is distributed according to the general distributions with sufficient min-entropy. The conclusion of this research is given in Section 6.4.

6.1 Preliminaries

The Standard LPN and Subspace LPN's definitions are presented in this section. The Standard LPN's definition is restated in term of the oracle machine. The Standard LPN's definition in this chapter seem to be different from LPN's definition in Chapter II but they are equivalents. Henceforth,

let the normal, bold, and capital bold letters like $x, \mathbf{x}, \mathbf{X}$ denote the single variables, vectors, and matrices, respectively. The symbol $x \xleftarrow{\$} X$ denotes that x is uniform sampled from the set X. Let $\mathcal{X}$ be the distribution, the symbol $x \leftarrow \mathcal{X}$ denotes sampling a value x with distribution $\mathcal{X}$.

Definition VI.1 *(Standard LPN)*
Let $n \in \mathbb{Z}^+$ be the security parameter, $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^n$ be the secret binary vector, and $\mathcal{X}$ be the error distribution over $\mathbb{Z}_2$. Let $\mathcal{O}(\mathbf{s}, \mathcal{X})$ be the oracle which takes $\mathbf{s}$ and $\mathcal{X}$ as the input parameters, and outputs the sample over $\mathbb{Z}_2^{n+1}$. Each sample is given by

$$(\mathbf{r}, \langle \mathbf{r}, \mathbf{s} \rangle \oplus e)$$

where $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_2^n$, and $e \leftarrow \mathcal{X}$. Normally, the error distribution $\mathcal{X}$ is Bernoulli distribution, which is denoted as Ber_η where $0 < \eta < 1/2$.

Definition VI.2 *(Hardness of Decisional LPN)*
The decisional LPN is (Q, ϵ) hard if for every PPT distinguisher D making the oracle queries $Q = poly(n)$ times,

$$\left| \Pr\left[D\left(\mathcal{O}^Q(\mathbf{s}, Ber_\eta) \right) = 1 \right] - \Pr\left[D\left(\mathcal{O}^Q(\mathbf{s}, U_2) \right) = 1 \right] \right| \leq \epsilon$$

where $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^n$, and U_2 is uniform distribution over $\mathbb{Z}_2$. Note that the oracle $\mathcal{O}(\mathbf{s}, U_2)$ outputs the uniform sample over $\mathbb{Z}_2^n \times \mathbb{Z}_2$.

The H_∞LPN's definition is the same as Standard LPN's definition but the secret vector is sampled from arbitrary distribution with min-entropy k instead of uniformly sampled from $\mathbb{Z}_2^n$.

Definition VI.3 *(Subspace LPN)*
Assume $\phi : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ is the affine function which is defined as $\phi(\mathbf{a}) = \mathbf{X} \cdot \mathbf{a} \oplus \mathbf{x}$, where $\mathbf{a} \in \mathbb{Z}_2^n$, $\mathbf{X} \in \mathbb{Z}_2^{n \times n}$, and $\mathbf{x} \in \mathbb{Z}_2^n$. Let $n \in \mathbb{Z}^+$ be the security parameter, $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^n$ be the secret binary vector, $\mathcal{X}$ be the error distribution over $\mathbb{Z}_2$, $\phi_r(\mathbf{r}) = \mathbf{X}_r \cdot \mathbf{r} \oplus \mathbf{x}_r$ be the affine projection of the random vector $\mathbf{r}$, and $\phi_s(\mathbf{s}) = \mathbf{X}_s \cdot \mathbf{s} \oplus \mathbf{x}_s$ be the affine projection of the secret vector $\mathbf{s}$. Let $\mathcal{Q}(\mathbf{s}, \mathcal{X}, k, \phi_r, \phi_s)$ be the oracle which outputs the sample over $\mathbb{Z}_2^{n+1}$ or null ($\varnothing$). The oracle $\mathcal{Q}$ will output

$$(\mathbf{r}, \langle \phi_r(\mathbf{r}), \phi_s(\mathbf{s}) \rangle \oplus e) \quad \textit{if} \quad \operatorname{rank}\left(\mathbf{X}_r^{\mathrm{T}} \mathbf{X}_s \right) \geq k$$

or output $\varnothing$; otherwise, where $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_2^n$, and $e \leftarrow \mathcal{X}$. The error distribution $\mathcal{X}$ is Bernoulli distribution as same as the Standard LPN.

Definition VI.4 *(Hardness of Decisional Subspace LPN)*
The decisional Subspace LPN is (Q, ϵ') *hard if for every PPT distinguisher* D' *making the oracle queries* $Q = poly(n)$ *times and can adaptively choosing* $\mathbf{X}_r$ *and* $\mathbf{X}_s$*,*

$$\left|\Pr\left[D'\left(\mathcal{Q}^Q\left(\mathbf{s}, Ber_\eta, k, \cdot, \cdot\right)\right) = 1\right] - \Pr\left[D'\left(\mathcal{Q}^Q\left(\mathbf{s}, U_2, k, \cdot, \cdot\right)\right) = 1\right]\right| \leq \epsilon'$$

where $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^n$*, and* U_2 *is uniform distribution over* $\mathbb{Z}_2$*. Note that, the oracle* $\mathcal{Q}\left(\mathbf{s}, U_2, \cdot, \cdot\right)$ *outputs the uniform sample over* $\mathbb{Z}_2^n \times \mathbb{Z}_2$ *if* $\mathrm{rank}\left(\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\right) \geq k$ *or outputs* $\varnothing$ *otherwise.*

Definition VI.5 *(min-entropy)*
The distribution $\mathcal{D}$ *has statistical min-entropy* k*, denoted as* $H_\infty\left(\mathcal{D}\right) = k$*, if for any* $x \leftarrow \mathcal{D}$*,* $\Pr\left[x\right] \leq 2^{-k}$.

6.2 The Hardness of H_∞LPN

In this section, the H_∞LPN is proof that it is as hard as Subspace LPN with security parameter $k < n$.

Theorem VI.1 H_∞*LPN, with security parameter* n *and the secret vector* $\mathbf{s}$ *is drawn from the distribution* $\mathcal{D}$ *with min-entropy* k*, is at least as hard as Subspace LPN, with* $\mathrm{rank}\left(\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\right) = k$.

proof: The proof of Theorem VI.1 is done by a reducing argument. The sample from the oracle $\mathcal{Q}\left(\mathbf{s}, Ber_\eta, k, \cdot, \cdot\right)$ is transformed to LPN sample with the secret vector $\mathbf{s} \in \mathbb{Z}_2^n$ is drawn from the distribution $\mathcal{D}$ with min-entropy k. This proof is relied on the following lemma.

Lemma VI.1 *Suppose* $\mathbf{M} \in \mathbb{Z}_2^{n \times n}$ *is the matrix with* $\mathrm{rank}\left(\mathbf{M}\right) = k$*, and* $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_2^n$ *is uniform binary vector of* n *dimensions. Then, the output vector* $\mathbf{Mv}$ *has min-entropy* k.

proof of Lemma VI.1: The matrix $\mathbf{M}$ has k maximum number of independent rows (or the maximum number of independent columns) since $\mathrm{rank}\left(\mathbf{M}\right) = k$. Without loss of generality, assume that $\mathbf{M}$ contains k independent rows. It is clearly see that the output vector $\mathbf{Mv}$ has independent k elements and the rest $n - k$ elements can be calculated from the k independent elements. Thus, for every $\mathbf{v} \in \mathbb{Z}_2^n$, the set of output vector $\mathbf{Mv}$ contains only 2^k different values. As a result, the result vector $\mathbf{Mv} \in \mathbb{Z}_2^n$ has min-entropy k. $\square$

The transform process is performed by querying the oracle $\mathcal{Q}\left(\mathbf{s}, \mathcal{X}, k, \cdot, \cdot\right)$ for Q times with parameters $\mathbf{X}_r = \mathbf{I}^{n \times n}, \mathbf{x}_r = \mathbf{x}_s = \mathbf{0}^n$, and $\mathbf{X}_s \in \mathbb{Z}_2^{n \times n}$ such that $\mathrm{rank}\left(\mathbf{X}_s\right) = k$. The oracle $\mathcal{Q}$ will not reply $\varnothing$ since $\mathrm{rank}\left(\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\right) \geq k$. It outputs the samples of the form

$$\left(\mathbf{r}, \langle \mathbf{r}, \mathbf{X}_s \cdot \mathbf{s} \rangle \oplus e\right) \quad \text{where} \quad \mathbf{r} \xleftarrow{\$} \mathbb{Z}_2^n, e \leftarrow \mathcal{X}$$

The vector $\mathbf{X}_s \cdot \mathbf{s}$ has min-entropy k, from Lemma VI.1, and obtains the H_∞LPN samples. Suppose that H_∞LPN is (Q, ϵ) hard, then there exist the distinguisher D such that

$$\left|\Pr\left[D\left(\mathcal{O}^{Q}\left(\mathbf{s}, Ber_\eta\right)\right) = 1 : \mathbf{s} \leftarrow \mathcal{D}\right] - \Pr\left[D\left(\mathcal{O}^{Q}\left(\mathbf{s}, U_2\right)\right) = 1 : \mathbf{s} \leftarrow \mathcal{D}\right]\right| \leq \epsilon.$$

Given the distinguisher D for solving the decisional H_∞LPN, one can construct the distinguisher D' for Subspace LPN by applying the transform process, then send the transformed samples to D. It is clearly see that the distinguisher D' has the advantage and the number of queries same as the distinguisher D. $\square$

Theorem VI.2 *Let $n \in \mathbb{Z}^+$ be the security parameter. Subspace LPN, with $\mathrm{rank}\left(\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\right) \geq k + \delta$ for some value of $\delta > 0$, is at least as hard as H_∞LPN, with secret vector $\mathbf{s}$ is drawn from the distribution $\mathcal{D}$ with min-entropy k.*

proof: This proof is shown by transforming the H_∞LPN sample from oracle $\mathcal{O}\left(\mathbf{s}, \mathcal{X}\right)$, where $\mathbf{s} \leftarrow \mathcal{D}$ has min-entropy k, to Subspace LPN sample from oracle $\mathcal{Q}\left(\mathbf{s}', \mathcal{X}, k + \delta, \cdot, \cdot\right)$, where $\mathbf{s}'$ is uniformly distributed over $\mathbb{Z}_2^n$.

Given $\mathbf{X}_r, \mathbf{X}_s \in \mathbb{Z}_2^{n \times n}$ such that $\mathrm{rank}\left(\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\right) \geq k + \delta$, $\mathbf{x}_r, \mathbf{x}_s \in \mathbb{Z}_2^n$, $\mathbf{M} \xleftarrow{\$} \mathbb{Z}_2^{n \times k}$, $\mathbf{N} \xleftarrow{\$} \mathbb{Z}_2^{k \times n}$, and $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_2^n$. Let $\mathcal{L}$ be the non-empty set of binary vectors in $\mathbb{Z}_2^n$ which is defined as follows

$$\mathcal{L} = \left\{\mathbf{y} : \mathbf{r}^{\mathrm{T}} = \mathbf{y}\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{M}\mathbf{N} \oplus \mathbf{x}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{M}\mathbf{N}\right\}$$

Suppose the oracle $\mathcal{O}\left(\mathbf{s}, \mathcal{X}\right)$ outputs the sample of the form $\left(\mathbf{r}, \langle \mathbf{r}, \mathbf{s}\rangle \oplus e\right)$, where $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_2^n, e \leftarrow \mathcal{X}$. It is transformed to the sample from $\mathcal{Q}\left(\mathbf{s}', \mathcal{X}, k + \delta, \cdot, \cdot\right)$ as follows

$$\left(\mathbf{r}', \langle \mathbf{r}, \mathbf{s}\rangle \oplus z \oplus e\right)$$

where $\mathbf{r}' \xleftarrow{\$} \mathcal{L}$, and $z = \left(\mathbf{X}_r\mathbf{r}' \oplus \mathbf{x}_r\right)^{\mathrm{T}} \cdot \left(\mathbf{X}_s\mathbf{v} \oplus \mathbf{x}_s\right)$. To proof that the transformed sample is the sample from $\mathcal{Q}\left(\mathbf{s}', \mathcal{X}, k + \delta, \cdot, \cdot\right)$ where $\mathbf{s}' = \mathbf{M}\mathbf{N}\mathbf{s} \oplus \mathbf{v}$, it is sufficient to prove the following lemmas.

Lemma VI.2 *Let $\mathcal{D}$ be a distribution over $\mathbb{Z}_2^n$ with sufficient high min-entropy k, $\mathbf{N} \xleftarrow{\$} \mathbb{Z}_2^{k \times n}$, and $\mathbf{s} \leftarrow \mathcal{D}$. The joint distribution of $(\mathbf{N}, \mathbf{N} \cdot \mathbf{s})$ is close to the uniform distribution over $\mathbb{Z}_2^{k \times n} \times \mathbb{Z}_2^k$.*

proof of Lemma VI.2: This research uses the matrix multiplication over $\mathbb{Z}_2$ as the universal hash function. From Generalized Leftover Hash Lemma [67], the distinguisher that can distinguishing $(\mathbf{N}, \mathbf{N} \cdot \mathbf{s})$ from $\mathbb{Z}_2^{k \times n} \times \mathbb{Z}_2^k$ has advantage at most $\epsilon' = \epsilon + \sqrt{\epsilon}$. $\square$

Lemma VI.3 *Let $\mathcal{D}$ be a distribution over $\mathbb{Z}_2^n$ with sufficient high min-entropy k. For $\mathbf{M} \xleftarrow{\$} \mathbb{Z}_2^{n \times k}$, $\mathbf{N} \xleftarrow{\$} \mathbb{Z}_2^{k \times n}$, $s \leftarrow \mathcal{D}$, and $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_2^n$, the vector $\mathbf{M}\mathbf{N}\mathbf{s} \oplus \mathbf{v}$ is close to the uniform distribution over $\mathbb{Z}_2^n$.*

proof of Lemma VI.3: From Lemma VI.2, the vector $\mathbf{Ns}$ is close to uniform distribution over $\mathbb{Z}_2^k$. Thus, $\mathbf{MNs} \oplus \mathbf{v}$ is close to the uniform distribution over $\mathbb{Z}_2^n$ according to standard LPN assumption with secret parameter k. $\square$

Lemma VI.4 *Assume* $\mathbf{X}_r, \mathbf{X}_s \in \mathbb{Z}_2^{n\times n}$, $\mathbf{x}_r \in \mathbb{Z}_2^n$, $\mathbf{M} \xleftarrow{\$} \mathbb{Z}_2^{n\times k}$, $\mathbf{N} \xleftarrow{\$} \mathbb{Z}_2^{k\times n}$, *and* $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_2^n$. *Let* $\mathcal{L}$ *be the non-empty set over* $\mathbb{Z}_2^n$ *such that* $\mathcal{L} = \left\{\mathbf{y} : \mathbf{r}^{\mathrm{T}} = \mathbf{y}\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{MN} \oplus \mathbf{x}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{MN}\right\}$. *If* $\mathrm{rank}\left(\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{MN}\right) \geq k$, *then the vector* $\mathbf{r}' \xleftarrow{\$} \mathcal{L}$ *is uniformly random.*

proof of Lemma VI.4 This lemma is proven by showing how to uniform sampling the vector r' from $\mathcal{L}$. Let $\mathbf{M}$ be the binary matrix, and $\mathbf{t}$ be the binary vector. The symbol $\mathbf{M}_{\downarrow\mathbf{t}}$ denotes the new matrix which is deleted the ith row if $\mathbf{t}[i] = 0$. There exist some $\mathbf{t}$ with hamming weight k such that $\left(\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{MN}\right)_{\downarrow\mathbf{t}}$ has full rank since $\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{MN}$ has rank at least k. The equation

$$\left(\mathbf{r}^{\mathrm{T}}\right)_{\downarrow\mathbf{t}} = \left(\mathbf{y}\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{MN} \oplus \mathbf{x}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{MN}\right)_{\downarrow\mathbf{t}} \tag{6.1}$$

has the unique solution of $\mathbf{y}_{\downarrow\mathbf{t}}$. The vector $\mathbf{r}'_{\downarrow\mathbf{t}} = \mathbf{y}_{\downarrow\mathbf{t}}$ is uniform distributed since $\mathbf{X}_r, \mathbf{X}_s, \mathbf{M}, \mathbf{N}$, and $\mathbf{x}_r$ are uniformly chosen.

The sampling method starts by solving Equation (6.1) to find the vector $\mathbf{r}'_{\downarrow\mathbf{t}}$. The rest $\mathbf{r}'_{\downarrow\bar{\mathbf{t}}}$'s elements are uniformly chosen. As a result, $\mathbf{r}'$ is uniform distributed over $\mathbb{Z}_2^n$. $\square$

Lemma VI.5 *Subspace LPN oracle* $\mathcal{Q}(\mathbf{s}', \mathcal{X}, k+\delta, \phi_r, \phi_s)$ *outputs the sample of the form*

$$\left(\mathbf{r}', \langle\phi_r(\mathbf{r}), \phi_s(\mathbf{s})\rangle \oplus e\right) = \left(\mathbf{r}', \left(\mathbf{r}'^{\mathrm{T}}\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{MN} \oplus \mathbf{x}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{MN}\right)\mathbf{s} \oplus \left(\mathbf{X}_r\mathbf{r}' \oplus \mathbf{x}_r\right)^{\mathrm{T}}\left(\mathbf{X}_s\mathbf{v} \oplus \mathbf{x}_s\right) \oplus e\right)$$

where $\mathbf{s}' = \mathbf{MNs} \oplus \mathbf{v}$.

proof of Lemma VI.5:

$$\begin{aligned}
\left(\mathbf{r}', \langle\phi_r(\mathbf{r}), \phi_s(\mathbf{s})\rangle \oplus e\right) &= \left(\mathbf{r}', \left(\mathbf{X}_r\mathbf{r}' \oplus \mathbf{x}_r\right)^{\mathrm{T}}\left(\mathbf{X}_s\mathbf{s}' \oplus \mathbf{x}_s\right) \oplus e\right) \\
&= \left(\mathbf{r}', \left(\mathbf{r}'^{\mathrm{T}}\mathbf{X}_r^{\mathrm{T}} \oplus \mathbf{x}_r^{\mathrm{T}}\right)\left(\mathbf{X}_s\left(\mathbf{MNs} \oplus \mathbf{v}\right) \oplus \mathbf{x}_s\right) \oplus e\right) \\
&= \left(\mathbf{r}', \left(\mathbf{r}'^{\mathrm{T}}\mathbf{X}_r^{\mathrm{T}} \oplus \mathbf{x}_r^{\mathrm{T}}\right)\left(\mathbf{X}_s\mathbf{MNs}\right) \oplus \left(\mathbf{r}'^{\mathrm{T}}\mathbf{X}_r^{\mathrm{T}} \oplus \mathbf{x}_r^{\mathrm{T}}\right)\left(\mathbf{X}_s\mathbf{v} \oplus \mathbf{x}_s\right) \oplus e\right) \\
&= \left(\mathbf{r}', \left(\mathbf{r}'^{\mathrm{T}}\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{MN} \oplus \mathbf{x}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{MN}\right)\mathbf{s} \oplus \left(\mathbf{X}_r\mathbf{r}' \oplus \mathbf{x}_r\right)^{\mathrm{T}}\left(\mathbf{X}_s\mathbf{v} \oplus \mathbf{x}_s\right) \oplus e\right)
\end{aligned}$$

$\square$

From Lemma VI.3, the secret vector $\mathbf{s}'$ is close to uniform distribution over $\mathbb{Z}_2^n$. The random vector $\mathbf{r}'$ is uniformly random according to Lemma VI.4. The output of Subspace LPN is shown in Lemma VI.5 and the vector $\mathbf{r}^{\mathrm{T}}$ is equal to vector $\mathbf{r}'^{\mathrm{T}}\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{MN} \oplus \mathbf{x}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{MN}$ since $\mathbf{r}'$ is chosen from

$\mathcal{L}$. As a result, the transform sample is valid because the sample from oracle $\mathcal{Q}\left(\mathbf{s}', \mathcal{X}, k+\delta, \phi_r, \phi_s\right)$ is of the form

$$\left(\mathbf{r}', \left(\mathbf{r}'^{\mathrm{T}}\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{M}\mathbf{N} \oplus \mathbf{x}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{M}\mathbf{N}\right)\mathbf{s} \oplus \left(\mathbf{X}_r\mathbf{r}' \oplus \mathbf{x}_r\right)^{\mathrm{T}}\left(\mathbf{X}_s\mathbf{v} \oplus \mathbf{x}_s\right) \oplus e\right) = \left(\mathbf{r}', \langle \mathbf{r}, \mathbf{s}\rangle \oplus z \oplus e\right)$$

where $\mathbf{r}' \xleftarrow{\$} \mathcal{L}$, and $z = \left(\mathbf{X}_r\mathbf{r}' \oplus \mathbf{x}_r\right)^{\mathrm{T}} \cdot \left(\mathbf{X}_s\mathbf{v} \oplus \mathbf{x}_s\right)$. Suppose that Subspace LPN is (Q, ϵ') hard, then there exist the distinguisher D' such that

$$\left|\Pr\left[D'\left(\mathcal{Q}^Q\left(\mathbf{s}, Ber_\eta, k, \cdot, \cdot\right)\right) = 1\right] - \Pr\left[D'\left(\mathcal{Q}^Q\left(\mathbf{s}, U_2, k, \cdot, \cdot\right)\right) = 1\right]\right| \leq \epsilon'$$

Given the distinguisher D' for solving the Subspace LPN, one can construct the distinguisher D for H_∞LPN by applying the transform process, then send the transformed samples to D'. However, the advantage of D is not the same as the advantage of D' because there is an error probability that the transformation will fail when $\mathrm{rank}\left(\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{M}\mathbf{N}\right) < k$. The following lemma will be used to calculate the error bound probability.

Lemma VI.6 *For $n, k, \delta \in \mathbb{Z}^+$, and $k+\delta \leq n$. Then,* $\Pr\left[\mathrm{rank}\left(\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{M}\mathbf{N}\right) < k\right] = \frac{2^k-1}{2^n}$.

proof of Lemma VI.6: This proof can be seen as the special case of the upper bound probability of random matrix in $\mathbb{Z}_q^{k\times k+\delta}$ which is proof in [66]. Assume that $\mathrm{rank}\left(\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{M}\mathbf{N}\right) < k$. Using matrix's rank property,

$$\mathrm{rank}\left(\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{M}\mathbf{N}\right) = \min\left\{\mathrm{rank}\left(\mathbf{M}\right), \mathrm{rank}\left(\mathbf{N}\right)\right\} < k$$

since $k \leq \mathrm{rank}\left(\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\right) = k+\delta$. Without loss of generality, assume that $\mathrm{rank}\left(\mathbf{M}\right) \leq \mathrm{rank}\left(\mathbf{N}\right)$, then

$$\Pr\left[\mathrm{rank}\left(\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{M}\mathbf{N}\right) < k\right] = \Pr\left[\min\left\{\mathrm{rank}\left(\mathbf{M}\right), \mathrm{rank}\left(\mathbf{N}\right)\right\} < k\right] = \Pr\left[\mathrm{rank}\left(\mathbf{M}\right) < k\right].$$

Recall that $\mathbf{M} \in \mathbb{Z}_2^{n\times k}$, and assume that $\mathbf{M}$ is generated by sampling each column one by one. Let E_i be the event that the first i columns are linearly independent. The probability that ith column is linearly dependent to the previous independent $i-1$ columns is $\Pr\left[\neg E_i | E_{i-1}\right] = \frac{2^{i-1}}{2^n}$. The probability that $\mathbf{M}$ contains rank less than k is calculated as follow

$$\Pr\left[\mathrm{rank}\left(\mathbf{M}\right) < k\right] = \sum_{i=1}^{k}\Pr\left[\neg E_i | E_{i-1}\right] = \sum_{i=1}^{k}\frac{2^{i-1}}{2^n} = \frac{2^k-1}{2^n}.$$

As a result, $\Pr\left[\mathrm{rank}\left(\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{M}\mathbf{N}\right) < k\right] = \Pr\left[\mathrm{rank}\left(\mathbf{M}\right) < k\right] = \frac{2^k-1}{2^n}$. □

For any Q oracle queries, the upper bound of probability that the chosen matrix $\mathbf{X}_r, \mathbf{X}_s, \mathbf{M}, \mathbf{N}$ will satisfy $\mathrm{rank}\left(\mathbf{X}_r^{\mathrm{T}}\mathbf{X}_s\mathbf{M}\mathbf{N}\right) < k$ is $\frac{\left(2^k-1\right)Q}{2^n}$, using Boole's inequality. The upper bound probability of the success distinguishing H_∞LPN is $\epsilon' - \frac{\left(2^k-1\right)Q}{2^n}$. For conclusion, the distinguisher D

for H_∞LPN, which is constructed using distinguisher D' for Subspace LPN, has advantage $\epsilon = \epsilon' - \frac{(2^k-1)Q}{2^n}$ and makes oracle query Q times. $\square$

From Theorem VI.1 and Theorem VI.2, the H_∞LPN is as hard as Subspace LPN. Thus, the H_∞LPN is as hard as standard LPN with security parameter k since Subspace LPN is as hard as standard LPN.

6.3 Applications/Discussions

The proofs in Theorem VI.1 and VI.2 show that the security parameter of LPN is reduced to min-entropy of the distribution that the secret vector is drawn. The loss of security parameter appears to be unavoidable since the secret vector has min-entropy only $k < n$. However, the H_∞LPN does not loss the noise distribution which is different form LWE [34]. In case of LWE, the standard deviation of error distribution α is reduced to β, where α/β has a negligible value. The loss of standard deviations causes the restriction on the modulus to be at least super-polynomial in n. The proofs imply that the cryptographic applications which are based on the LPN assumption are robustness although the secret key is drawn from general distributions of high min-entropy.

This work also proves that the probabilistic CPA symmetric key encryption scheme in previous work [50] is secure even if the secret key is sampling from an arbitrary distribution with sufficient min-entropy. This scheme is defined as $\mathbb{E} = (K, E, D)$ such that:

- **Parameters**. Security parameter n, and noise level $0 < \eta < 1/2$.
- **Public Components**. An error-correcting code C and the corresponding decoding algorithm C^{-1}.
- **Secret Key Generation** K. On input 1^n, outputs a uniform random secret key $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^n$.
- **Encryption Algorithm** E. On input a secret key $\mathbf{s}$ and a message $\mathbf{p} \in \mathbb{Z}_2^m$,

  $$E_{\mathbf{s}}(\mathbf{p}) = (\mathbf{A}, \mathbf{As} \oplus \mathbf{e} \oplus C(\mathbf{p}))$$

  where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{m \times n}$ and $\mathbf{e} \xleftarrow{\$} Ber_\eta^m$.
- **Decryption Algorithm** D. On input a secret key $\mathbf{s}$ and a ciphertext $(\mathbf{A}, \mathbf{y})$,

  $$D_{\mathbf{s}}((\mathbf{A}, \mathbf{y})) = C^{-1}(\mathbf{As} \oplus \mathbf{y}).$$

This scheme was proven that it is CPA secure and key hiding property w.r.t. exponentially hard-to-invert auxiliary input [50]. This research proves the further security about the robustness of

this scheme when the secret key is distributed according to an arbitrary distribution with sufficient min-entropy. Note that, the definition of CPA secure w.r.t $k\left(n\right)$-weak keys was well defined in [34].

Theorem VI.3 *For any distribution $\mathcal{D}=\{\mathcal{D}_n\}_{n\in\mathbb{N}}$ with min-entropy $k=k\left(n\right)$, let $K_{\mathcal{D}}\left(1^n\right)$ be the Secret Key Generation algorithm which outputs the secret key $\mathbf{s}\xleftarrow{\$}\mathcal{D}_n$. Then, the encryption scheme $\mathbb{E}=\left(K_{\mathcal{D}},E,D\right)$ is CPA secure w.r.t. $k\left(n\right)$-weak keys, under the H$_\infty$LPN assumption.*

proof: Let $\mathcal{E}_{\mathbf{s},\mathcal{X}}\left(\mathbf{p}\right)$ be the encryption oracle that takes message $\mathbf{p}$ as an input, then output the sample of the form

$$\left(\mathbf{A},\mathbf{As}\oplus\mathbf{e}\oplus C\left(\mathbf{p}\right)\right)$$

where $\mathbf{A}\xleftarrow{\$}\mathbb{Z}_2^{m\times n}$, and $\mathbf{e}\leftarrow\mathcal{X}^m$. The oracle $\mathcal{E}$ outputs the ciphertext sample when the distribution $\mathcal{X}$ is Ber_η^m or outputs uniform sample over $\mathbb{Z}_2^m$ when the distribution $\mathcal{X}$ is U_2^m. This theorem is proven by showing that there is no PPT adversary algorithm A, which can query the oracle $\mathcal{E}$ for $Q=poly\left(n\right)$ times, such that

$$\left|\Pr\left[A\left(\mathcal{E}^Q_{\mathbf{s},Ber_\eta^m}\left(\mathbf{p}\right)\right)=1\right]-\Pr\left[A\left(\mathcal{E}^Q_{\mathbf{s},U_2^m}\left(\mathbf{p}\right)\right)=1\right]\right|>\epsilon\left(n\right)$$

where $\mathbf{s}\leftarrow\mathcal{D}_n$, and $\epsilon\left(n\right)$ is not a negligible function. Suppose there exist the PPT adversary A, it can be used to construct the distinguisher D for H$_\infty$LPN such that

$$\left|\Pr\left[D\left(\mathcal{O}^{m\times Q}\left(\mathbf{s},Ber_\eta\right)\right)=1\right]-\Pr\left[D\left(\mathcal{O}^{m\times Q}\left(\mathbf{s},U_2\right)\right)=1\right]\right|>\epsilon\left(n\right)$$

where $\mathbf{s}\leftarrow\mathcal{D}_n$. The algorithm D can be constructed by calling the adversary A. Every time A asks for querying the oracle $\mathcal{E}$ on secret key $\mathbf{s}$, message $\mathbf{p}$, and noise distribution $\mathcal{X}^m$, the distinguisher D simply queries the oracle $\mathcal{O}\left(\mathbf{s},\mathcal{X}\right)$ for m times. Then, the samples are compacted in form of $\left(\mathbf{A},\mathbf{y}\right)$ and it is send to adversary A, where $\mathbf{A}\in\mathbb{Z}_2^{m\times n}$ and $\mathbf{y}\in\mathbb{Z}_2^m$. Each row of matrix $\mathbf{A}$ is the vector $\mathbf{r}$ and each element of vector $\mathbf{y}$ is $\langle\mathbf{r},\mathbf{s}\rangle\oplus e\oplus\mathbf{p}\left[i\right]$, where $e\leftarrow\mathcal{X}$. As a result, there exist PPT distinguisher D which can distinguishing H$_\infty$LPN samples with non-negligible advantage that contradicts to the H$_\infty$LPN assumption. □

6.4 Conclusions

This chapter proves the open question about the hardness of LPN when the secret vector $\mathbf{s}\in\mathbb{Z}_2^n$ is sampling from an arbitrary distribution $\mathcal{D}$ with min-entropy k. For any $k<n$, the proofs show that security parameter is reduced from n to k when the secret vector has only k min-entropy. Formally, if the secret vector has min-entropy k, then the security parameter of LPN assumption is roughly k. The

robustness of LPN implies that the cryptographic primitives that are constructed based on the LPN assumption still secure even if the secret vector has min-entropy $k < n$ but the security parameter is reduced to min-entropy itself. This work shows that, under the LPN assumption, the symmetric-key encryption scheme in the previous work is secure even if the secret key has min-entropy only k. The robustness of LPN can be applied to other cryptographic applications, such as HB identification protocol, pseudorandom generator, and so on.

CHAPTER VII

RESULTS AND DISCUSSIONS

In this chapter, the overall results of each topic in this dissertation are described in Section 7.1, 7.2, 7.3, and 7.4.

7.1 Results and Discussions on Weight-adjusting Strong Distinguishing attack

Many strong distinguishing attacks rely on Pearson's chi-square hypothesis test. The number of samples which is used to distinguish the pseudorandom digits from the uniform distribution usually estimated by the rule of thumb. This approximation does not always yield the optimal number of samples. Moreover, they lack the method for verifying the correctness level when the number of samples is limited due to the restrict resources environment. The weight-adjusting strong distinguishing attack framework can solve these problems. The precise number of samples can be calculated to obtain enough advantage and avoid the overestimate problem. Moreover, in the situation that the attacker has inadequate samples, the proposed method is the best option since it is based on Binomial hypothesis test.

However, the proposed method is valid only when the based weak distinguisher has specific properties. Although any weak distinguishers can be transformed to the specific properties weak distinguisher, the weak distinguisher's advantage is also reduced by some constants. This condition causes the strong distinguisher requires more samples. Nevertheless, the analysis of stream ciphers's biases is mostly done on the less significant or most significant bit of the stream cipher's outputs. This analysis makes the weak distinguisher has specific properties as required in the weight-adjusting strong distinguisher.

7.2 Results and Discussions on the RC4B

In order to avoid the general distinguishing attack on array based stream cipher and to preserve the stream cipher's performance, the hybrid random generator between RC4A and Blum-Blum-Shub is proposed in this thesis, the RC4B. The RC4B is proven that it gains higher performance than Blum-Blum-Shub and has higher security than RC4A. Unfortunately, RC4B is not cryptographically

secure as Blum-Blum-Shub since RC4A's permutation algorithm is not perfectly permuted its arrays. This thesis also shows the permutation bias on RC4A which can affect to the RC4B's outputs. The analysis shows that the RC4B's distinguisher needs larger samples than the RC4A's distinguisher. The number of samples in the RC4B is exponentially increased by the factor of 2^l. This distinguishing attack cannot be done in practice when the primes are large enough.

7.3 Results and Discussions on LWE-based Pseudorandom Generator

The LWE-based PRG is constructed based on the hardness assumption of Regev's LWE problem, which is believed that it is extremely hard even quantum computers cannot efficiently solve. The problem of distinguishing the LWE-based PRG's outputs from the uniform distribution is reducible to Regev's LWE problem. This proof implies that the distinguishing problem obtains the hardness from Regev's LWE problem. Differently, the traditional pseudorandom generators mostly depended on the hardness of Factoring assumption, such as Blum-Blum-Shub Generator, RSA generator, and Blum-Micali Generator. The integer factorization problem can be solved in the polynomial time, using Shor's quantum algorithm [68], [69]. Thus, the LWE-based PRG gains higher security over the traditional pseudorandom generators.

The drawback of LWE-based PRG is the amount of randomness which is required to initiate the generator. The vector's dimension should be at least 60 since the best known algorithm for solving LWE problem takes $2^{O(n)}$ time complexity. Therefore, it requires at least $7,260$ uniform random numbers over $\mathbb{Z}_q$ for filling the public matrix, $\mathbf{A}$, and the secret vector, $\mathbf{x}$. The another reason of excessive randomness input is the lacks of efficient algorithm for sampling the Gaussian noise from the short seed. The Gaussian noise generator normally requires the uniform seeds as the inputs. For example, Box-Muller transform takes a pair of uniform random numbers over continuous range $(0, 1)$ and generates a pair of Gaussian random number with mean $\mu = 0$ and variance $\sigma^2 = 1$. The best known algorithm for generating the Gaussian noise has ratio between the number of uniform distributed inputs and the number of Gaussian distribution outputs is 1 [70]. The method to overcome this problem is to generate sufficient uniform random numbers from another pseudorandom generator and send them to Gaussian random number Generator.

7.4 Results and Discussions on the Robustness of Learning Parity with Noise Assumption

The traditional cryptosystems mostly assume that there is no information about the secret key is leaked. However, in the real word situation, there are many methods to obtain the secret key's information, such as Power Analysis Attacks, Memory Attacks, Timing Attacks, and Fault Injection Attacks. These types of attacks are called Side Channel Attacks. One of the important questions in the cryptographic research topic is

"How to design the cryptosystem that remains secure even if the secret key is leaked?"

The good leakage resilient cryptosystem should be relied on the cryptographic assumption that holds even in the arbitrary key-leakage environment. The Learning with Errors assumption was proven that it is robustness. However, the proof does not mean anything about its special case, the Learning Parity with Noise assumption, since Learning with Errors require the prime q that must be the super-polynomial of secret vector's dimension n.

This thesis shows that the Learning Parity with Noise assumption is robust to the leaky secret key. Formally, the Learning Parity with Noise assumption remains secure even if the secret key has min-entropy $k < n$, where n is the number of secret vector's dimensions. The proof shows that the security parameter is reduced to min-entropy itself but it keeps the noise distribution untouched. This is different from the case of Learning with Errors which blows up the noise distribution and causes the prime must be large. Based on the robustness of Learning Parity with Noise, many fortitude cryptosystems can be constructed. For examples, the pseudorandom generators that still secure even if the seed is not uniform distributed.

CHAPTER VIII

CONCLUSIONS AND FUTURE WORKS

This thesis mainly concerns about theoretical aspects of the pseudorandom generator. Initially, the analysis on the strong distinguishing attack is presented in Chapter III. The alternative distinguishing approach provides more accurate method for estimating the number of samples. The space and time complexity are optimized since the number of samples can be adjusted to match the resource. In order to improve the security of the array based stream cipher, the hybrid stream cipher, the RC4B, is proposed in Chapter IV. The analysis shows that the hybrid method can trade-off between the security and the execution time. If the security is the most important issue, the RC4B's parameters can be adapted for gaining high security level as Blum-Blum-Shub generator. On the other hand, the security of RC4B can be relaxed when the performance is an important issue. The extremely secure pseudorandom generator, the LWE-based PRG, is proposed in Chapter V. The LWE-based PRG is proven to be secure against any attacks by the quantum computer. This research also studies the hardness of Learning Parity with Noise assumption when the secret key is not uniform distributed. The robustness of Learning Parity with Noise is proven in Chapter VI. The result shows that the Learning Parity with Noise assumption remains secure even the secret key is non-uniform distributed but the security parameter is reduced to the secret key's min-entropy itself.

In this chapter, the overall conclusions of each topic in this dissertation are described in Section 8.1, 8.2, 8.3, and 8.4. The future works are also suggested in Section 8.5.

8.1 Conclusions on Weight-Adjusting Strong Distinguisher

The traditional strong distinguishing attack is normally based on Pearson's chi-square hypothesis testing. It uses several samples to increase the correctness level. The number of samples is usually estimated to $1/\varepsilon^2$, where ε is the random bits's bias. However, it lacks the method for verifying the accurate strong distinguisher's advantage, given the number of samples. It may not obtain high advantage as expected if the number of samples is underestimated. This research proposes the new frameworks for strong distinguisher, which is solely based on binomial hypothesis testing instead of Pearson's chi-square test. The proposed method can calculate the exactly number of samples which can satisfy the expected correctness level. It can verify the obtained advantage when the memory is

limited. This work also shows that the commonly used estimation method always undervalues the number of samples. As a result, the alternative approach strong distinguisher can adjust the number of samples for matching the memory's size and can show the obtained correctness level.

8.2 Conclusions on the RC4B

The stream cipher is a symmetric key cryptosystem where the plaintext is combined with the keystream to produce a ciphertext. It is often used because of speed and simplicity. However, the keystream from a stream cipher is not real-random and can be distinguished after examining some amount of its random stream. In contrast, the cryptographic secure pseudorandom generator produces the random stream which cannot be distinguished in the polynomial time but it requires a long execution time and cannot be used to generate a large amount of keystreams. To trade-off between efficiency and security, the hybrid stream cipher, the RC4B, is proposed in this research. The proposed method shows that it gains much security over the based stream cipher, RC4A, and require less execution time than the based cryptographic secure pseudorandom generator, Blum-Blum-Shub Generator. Moreover, this research also proposed the new statistical bias which can affect the prototype stream cipher and the hybrid generator. The proof shows that this bias causes less effect on the hybrid algorithm and the required number of samples is exponentially growth by factor of 2^l.

8.3 Conclusions on LWE-based Pseudorandom Generator

The construction and security proof of pseudorandom generator based on Learning with Errors problem, the LWE-based PRG, is presented in this dissertation. The Learning with Errors problem is believed that it cannot be efficiently solved by any algorithms, even quantum algorithms. It was used as a tool for constructing many cryptographic applications, such as public-key cryptosystem, leakage-resilient encryption, cryptographic protocols, and so on. The LWE-based PRG is easy to be implemented and contains many parallel steps. This thesis shows that the problem of distinguishing LWE-based PRG's random outputs from uniform random sequences is at least as hard as solving Regev's Learning with Errors Problem. Therefore, there is no efficient algorithm, including quantum algorithm, for attacking LWE-based PRG's outputs.

8.4 Conclusions on the Robustness of Learning Parity with Noise Assumption

The classical cryptographic primitives are constructed on the assumption that the private key is kept secret and uniformly distributed. Learning Parity with Noise is the famous problem which

is used to construct several cryptographic primitives. This research studies the open question about the hardness of the Learning Parity with Noise assumption when the secret vector is not uniform and has min-entropy only k. The proof shows that the standard Learning Parity with Noise implies that it is secure even if the secret vector is sampled from an arbitrary distribution with sufficient entropy. Furthermore, this research shows that the symmetric encryption scheme from LPN is secure even if the secret key has min-entropy k.

8.5 Future Works

The future works are listed as follow:

- The framework for the strong distinguishing attack based on general properties weak distinguishing attack.
- Improving the efficiency of LWE-based PRG and constructing the method for sampling the noise vector.
- The robustness cryptographic primitives based on the H_∞LPN that remain secure even the secret key is non-uniform distributed.

REFERENCES

[1] Shannon, C. E. Communication Theory of Secrecy Systems. Bell System Technical Journal, Vol 28, pp. 656–715.

[2] Shannon, C. E. A mathematical theory of communication. SIGMOBILE Mob. Comput. Commun. Rev. 5 (Jan. 2001): 3–55.

[3] Goldreich, O. A Note on Computational Indistinguishability. Inf. Process. Lett. 34 , 6 (1990): 277-281.

[4] Yao, A. C.-C., Theory and Applications of Trapdoor Functions (Extended Abstract). FOCS, IEEE Computer Society, (1982): 80-91.

[5] Goldwasser, S. and Micali, S. Probabilistic Encryption. J. Comput. Syst. Sci. 28 , 2 (1984): 270-299.

[6] Biham, E. and Seberry, J. Py (Roo): A Fast and Secure Stream Cipher using Rolling Arrays. IACR Cryptology ePrint Archive 2005 (2005): 155.

[7] Eli Biham, J. S., Pypy: Another Version of Py., eSTREAM, ECRYPT Stream Cipher Project, Report 2006/038 (2006).

[8] Eli Biham, J. S., Tweaking the IV Setup of the Py Family of Stream Ciphers – The Ciphers TPy, TPypy, and TPy6., eSTREAM, ECRYPT Stream Cipher Project, Report 2007/038 (2007).

[9] Wu, H., A New Stream Cipher HC-256. in Roy and Meier [71], pp. 226–244.

[10] Ferguson, N., Whiting, D., Schneier, B., Kelsey, J., Lucks, S., and Kohno, T., Helix: Fast Encryption and Authentication in a Single Cryptographic Primitive. FSE (Johansson, T., ed.) 2887 (2003): 330-346.

[11] Daemen, J., Kitsos, P., and Belgium, J. D. S., Submission to ECRYPT call for stream ciphers: the self-synchronizing stream cipher MOSQUITO., 2005.

[12] Daemen, J. and Kitsos, P., New Stream Cipher Designs. ch. The Self-synchronizing Stream Cipher Moustique, pp. 210–223, Berlin, Heidelberg : Springer-Verlag, 2008.

[13] Philip Hawkes, G. G. R. M. W. d. V., Primitive Specification for SSS., 2005.

[14] Andrew Rukhin, J. N. M. S. E. B. S. L. M. L. M. V. D. B. A. H. J. D. S. V., A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications.

[15] Matsui, M., Linear Cryptoanalysis Method for DES Cipher. EUROCRYPT (Helleseth, T., ed.) 765 (1993): 386-397.

[16] Golic, J. D., Linear Cryptanalysis of Stream Ciphers. FSE (Preneel, B., ed.) 1008 (1994): 154-169.

[17] Golic, J. D., Bagini, V., and Morgari, G., Linear Cryptanalysis of Bluetooth Stream Cipher. EUROCRYPT (Knudsen, L. R., ed.) 2332 (2002): 238-255.

[18] Cho, J. Y. and Pieprzyk, J., Distinguishing Attack on SOBER-128 with Linear Masking. ACISP (Batten, L. M. and Safavi-Naini, R., eds.) 4058 (2006): 29-39.

[19] Watanabe, D., Biryukov, A., and Cannière, C. D., A Distinguishing Attack of SNOW 2.0 with Linear Masking Method. Selected Areas in Cryptography (Matsui, M. and Zuccherato, R. J., eds.) 3006 (2003): 222-233.

[20] Nyberg, K. and Wallén, J., Improved Linear Distinguishers for SNOW 2.0. FSE (Robshaw, M. J. B., ed.) 4047 (2006): 144-162.

[21] Gong, G., Gupta, K. C., Hell, M., and Nawaz, Y., Towards a General RC4-Like Keystream Generator. CISC (Feng, D., Lin, D., and Yung, M., eds.) 3822 (2005): 162-174.

[22] Zoltak, B., VMPC One-Way Function and Stream Cipher. in Roy and Meier [71], pp. 210–225.

[23] Paul, S. and Preneel, B., A New Weakness in the RC4 Keystream Generator and an Approach to Improve the Security of the Cipher. in Roy and Meier [71], pp. 245–259.

[24] Scream: A Software-Efficient Stream Cipher.. Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers, (2002): 195-209.

[25] A New Keystream Generator MUGI.. Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers, (2002): 179-194.

[26] Paul, S. and Preneel, B., On the (In)security of Stream Ciphers Based on Arrays and Modular Addition. ASIACRYPT (Lai, X. and Chen, K., eds.) 4284 (2006): 69-83.

[27] Englund, H., Johansson, T., and Turan, M. S., A Framework for Chosen IV Statistical Analysis of Stream Ciphers. INDOCRYPT (Srinathan, K., Rangan, C. P., and Yung, M., eds.) 4859 (2007): 268-281.

[28] O. Saarinen, M., Chosen-iv statistical attacks on estream stream ciphers. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/013, (2006): 5–19.

[29] Regev, O., On lattices, learning with errors, random linear codes, and cryptography. STOC (Gabow, H. N. and Fagin, R., eds.), ACM, (2005): 84-93.

[30] Regev, O. On lattices, learning with errors, random linear codes, and cryptography. J. ACM 56 , 6.

[31] Peikert, C. Public-Key Cryptosystems from the Worst-Case Shortest Vector Problem. Electronic Colloquium on Computational Complexity (ECCC) 15 , 100.

[32] Blum, A., Kalai, A., and Wasserman, H. Noise-tolerant learning, the parity problem, and the statistical query model. J. ACM 50 , 4 (2003): 506-519.

[33] Arora, S. and Ge, R., New Algorithms for Learning in Presence of Errors. ICALP (1) (Aceto, L., Henzinger, M., and Sgall, J., eds.) 6755 (2011): 403-415.

[34] Goldwasser, S., Kalai, Y. T., Peikert, C., and Vaikuntanathan, V., Robustness of the Learning with Errors Assumption. ICS (Yao, A. C.-C., ed.), Tsinghua University Press, (2010): 230-240.

[35] Dodis, Y., Goldwasser, S., Kalai, Y. T., Peikert, C., and Vaikuntanathan, V., Public-Key Encryption Schemes with Auxiliary Inputs. TCC (Micciancio, D., ed.) 5978 (2010): 361-381.

[36] Banerjee, A., Peikert, C., and Rosen, A. Pseudorandom Functions and Lattices. IACR Cryptology ePrint Archive 2011 (2011): 401.

[37] Peikert, C., Vaikuntanathan, V., and Waters, B., A Framework for Efficient and Composable Oblivious Transfer. CRYPTO (Wagner, D., ed.) 5157 (2008): 554-571.

[38] Lyubashevsky, V., Micciancio, D., Peikert, C., and Rosen, A., SWIFFT: A Modest Proposal for FFT Hashing. FSE (Nyberg, K., ed.) 5086 (2008): 54-72.

[39] Agrawal, S., Boneh, D., and Boyen, X., Efficient Lattice (H)IBE in the Standard Model. in Gilbert [72], pp. 553–572.

[40] Cash, D., Hofheinz, D., Kiltz, E., and Peikert, C., Bonsai Trees, or How to Delegate a Lattice Basis. in Gilbert [72], pp. 523–552.

[41] Gentry, C., Peikert, C., and Vaikuntanathan, V., Trapdoors for hard lattices and new cryptographic constructions. STOC (Dwork, C., ed.), ACM, (2008): 197-206.

[42] Berlekamp, E. R., Mceliece, R. J., and Van Tilborg, H. C. A. On the inherent intractability of certain coding problems. IEEE Transactions on Information Theory 24.

[43] Katz, J., Shin, J. S., and Smith, A. Parallel and Concurrent Security of the HB and HB^+ Protocols. J. Cryptology 23 , 3 (2010): 402-421.

[44] Fossorier, M. P. C., Mihaljevic, M. J., Imai, H., Cui, Y., and Matsuura, K., An Algorithm for Solving the LPN Problem and Its Application to Security Evaluation of the HB Protocols for RFID Authentication. INDOCRYPT (Barua, R. and Lange, T., eds.) 4329 (2006): 48-62.

[45] Levieil, É. and Fouque, P.-A., An Improved LPN Algorithm. SCN (Prisco, R. D. and Yung, M., eds.) 4116 (2006): 348-359.

[46] Wagner, D., A Generalized Birthday Problem. CRYPTO (Yung, M., ed.) 2442 (2002): 288-303.

[47] Blum, A., Furst, M. L., Kearns, M. J., and Lipton, R. J., Cryptographic Primitives Based on Hard Learning Problems. CRYPTO (Stinson, D. R., ed.) 773 (1993): 278-291.

[48] Applebaum, B., Cash, D., Peikert, C., and Sahai, A., Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. CRYPTO (Halevi, S., ed.) 5677 (2009): 595-618.

[49] Gilbert, H., Robshaw, M. J. B., and Seurin, Y., How to Encrypt with the LPN Problem. ICALP (2) (Aceto, L., Damgård, I., Goldberg, L. A., Halldórsson, M. M., Ingólfsdóttir, A., and Walukiewicz, I., eds.) 5126 (2008): 679-690.

[50] Dodis, Y., Kalai, Y. T., and Lovett, S., On cryptography with auxiliary input. STOC (Mitzenmacher, M., ed.), ACM, (2009): 621-630.

[51] Hopper, N. J. and Blum, M., Secure Human Identification Protocols. ASIACRYPT (Boyd, C., ed.) 2248 (2001): 52-66.

[52] Juels, A. and Weis, S. A., Authenticating Pervasive Devices with Human Protocols. CRYPTO (Shoup, V., ed.) 3621 (2005): 293-308.

[53] Applebaum, B., Barak, B., and Wigderson, A., Public-key cryptography from different assumptions. in Schulman [73], pp. 171–180.

[54] Maximov, A. Two Linear Distinguishing Attacks on VMPC and RC4A and Weakness of RC4 Family of Stream Ciphers (Corrected). IACR Cryptology ePrint Archive 2007 (2007): 70.

[55] Tsunoo, Y., Saito, T., Kubo, H., Shigeri, M., Suzaki, T., and Kawabata, T. The Most Efficient Distinguishing Attack on VMPC and RC4A.

[56] Blum, L., Blum, M., and Shub, M. A Simple Unpredictable Pseudo-Random Number Generator. SIAM J. Comput. 15 , 2 (1986): 364-383.

[57] Blum, L., Blum, M., and Shub, M., Comparison of Two Pseudo-Random Number Generators. CRYPTO (Chaum, D., Rivest, R. L., and Sherman, A. T., eds.), Plenum Press, New York, (1982): 61-78.

[58] Sidorenko, A. and Schoenmakers, B., Concrete Security of the Blum-Blum-Shub Pseudorandom Generator. IMA Int. Conf. (Smart, N. P., ed.) 3796 (2005): 355-375.

[59] Cusick, T. W. Properties of the x^2 mod N pseudorandom number generator. IEEE Transactions on Information Theory 41 , 4 (1995): 1155-1159.

[60] Joye, M. and Paillier, P., Fast Generation of Prime Numbers on Portable Devices: An Update. CHES (Goubin, L. and Matsui, M., eds.) 4249 (2006): 160-173.

[61] Mantin, I. and Shamir, A., A Practical Attack on Broadcast RC4. FSE (Matsui, M., ed.) 2355 (2001): 152-164.

[62] Hoffstein, J., Pipher, J., and Silverman, J. H., NTRU: A Ring-Based Public Key Cryptosystem. ANTS (Buhler, J., ed.) 1423 (1998): 267-288.

[63] Lyubashevsky, V., Peikert, C., and Regev, O., On Ideal Lattices and Learning with Errors over Rings. in Gilbert [72], pp. 1–23.

[64] Stehlé, D., Steinfeld, R., Tanaka, K., and Xagawa, K., Efficient Public Key Encryption Based on Ideal Lattices. ASIACRYPT (Matsui, M., ed.) 5912 (2009): 617-635.

[65] Pietrzak, K., Cryptography from Learning Parity with Noise. SOFSEM (Bieliková, M., Friedrich, G., Gottlob, G., Katzenbeisser, S., and Turán, G., eds.) 7147 (2012): 99-114.

[66] Pietrzak, K., Subspace LWE. TCC (Cramer, R., ed.) 7194 (2012): 548-563.

[67] Barak, B., Dodis, Y., Krawczyk, H., Pereira, O., Pietrzak, K., Standaert, F.-X., and Yu, Y., Leftover Hash Lemma, Revisited. CRYPTO (Rogaway, P., ed.) 6841 (2011): 1-20.

[68] Shor, P. W., Polynominal time algorithms for discrete logarithms and factoring on a quantum computer. ANTS (Adleman, L. M. and Huang, M.-D. A., eds.) 877 (1994): 289.

[69] Shor, P. W., Algorithms for Quantum Computation: Discrete Logarithms and Factoring. FOCS, IEEE Computer Society, (1994): 124-134.

[70] Thomas, D. B., Luk, W., Leong, P. H. W., and Villasenor, J. D. Gaussian random number generators. ACM Comput. Surv. 39 , 4.

[71] Roy, B. K. and Meier, W., eds., Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers, vol. 3017 of *Lecture Notes in Computer Science*, Springer, 2004.

[72] Gilbert, H., ed., Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings, vol. 6110 of *Lecture Notes in Computer Science*, Springer, 2010.

[73] Schulman, L. J., ed., Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010, ACM, 2010.

Biography

Education:

- Ph.D. Program in Computer Science, Faculty of Science, Chulalongkorn University, Thailand (June 2007 - February 2012).
- B.Sc. Program in Computer Science, Faculty of Science, Chulalongkorn University, Thailand (June 2003 - February 2006).

Publication Papers:

- V. Suttichaya and P. Bhattarakosol, "Enhancing Cryptographic Secure PRBG by Appling Permutation on $x^2 \bmod n$ Generator," in *Proc. 2009 Int. Conf. on Wireless Information Networks & Business Information System (WINBIS 09)*, Kathmandu, Nepal, February 27- March 1, 2009.
- V. Suttichaya and P. Bhattarakosol, "How to Obtain High Security Over the E-Commerce System," in *International Conference on e-Business (INCEB2008)*, Bangkok, Thailand, 2008.
- V. Suttichaya and P. Bhattarakosol, "Chain Rule Protection over the Internet Using PUGGAD Algorithm," in *Proc. Int. Conf. on Computer and Electrical Engineering (ICCEE2008)*, Phuket, Thailand, December 20-22, pp. 492-496, 2008.

Scholarships:

- Office of the Higher Education Commission, Thailand (June 2008 - May 2012).
- Chulalongkorn University Graduate Scholarship to Commemorate the 72nd Anniversary of His Majesty King Bhumibol Adulyadej (June 2007 - May 2008).